Network Communication Protocols for High School Students

October 17, 2014

Samuel Jarman
saj77@uclive.ac.nz
Department of Computer Science and Software Engineering
University of Canterbury, Christchurch, New Zealand

Supervisor: Professor Tim Bell
Tim.Bell@canterbury.ac.nz

Abstract

Computer Science was introduced to New Zealand schools as an NCEA topic for the first time in 2011. Development of resources for both teachers and students has been crucial in the topic's success. The Computer Science Field Guide is an open source, online textbook aimed at solving this problem. This project describes the design and development of the final chapter of the textbook on network protocols. The textbook includes a video, classroom activities, online games, and a tutorial, which are made up on existing resources that have been improved and new content which has been developed as needed. The content covers what a protocol is, what they can achieve and how protocols can be layered to achieve suitable abstractions. Through a pilot study, metrics and teacher feedback, we are confident the content will be useful and engaging for students studying Computer Science at Level 3.

Acknowledgments

A special thank you to my supervisor, Professor Tim Bell, my colleagues in the computer science education research group, my friends and family.

Also thanks to the New Zealand Association for Computing, Digital and Information Technology Teachers (NZACDITT) and The College of Education for letting me be involved and gain valuable insights from teachers directly.

Contents

1 Introduction

1.1 Background

In a world becoming more and more digital, new technologies, companies and startups are being created every day. To support this, the industry needs people with the right skill set and talent. With the birth of new companies, more and more jobs are on offer for Information Communication Technology (ICT) graduates. In 2006, Brislen reported that by 2010, there would be 66,000 roles available in New Zealand, and in the UK, 150,000 roles, noting that the number of roles were very high globally [7]. However, there simply are not enough graduates coming from tertiary education and these roles are not being filled [8, 10]. Brislen reported that from 2006 to 2010, New Zealand would only produce 3,500 graduates, and the UK only 20,000 graduates, a staggeringly small fraction of what is required. The industry is feeling the strain and something has to be done in order for our society to be able to keep moving forward with our digital world aspirations.

From this, it is clear to see that getting more students in the field and graduating is important. However, the interest in the field, specifically Computer Science (CS) starts earlier than tertiary, it starts in High School. This is not happening. There are a few reasons why students may not be interested in CS, and the most observed one of these is a lack of exposure to the field, and subsequently students were not confident in choosing CS as a career. Another big reason is an incorrection perception — students "don't really understand what Computer Science is"[1] and this was also noted in "Unlocking the Clubhouse" [20]. A negative introduction by unconfident or unenthused teachers is also a negative factor for students choosing CS. What is needed is a positive introduction to the field by confident teachers.

Realising these issues existed, in 2008, reports were published by both the Ministry of Education and the Institute of Information Technology Professionals [18, 9]. These reports revealed a mistaken, poorly thought out plan for CS in high schools. Originally, high schools only offered "computing" (sometimes referred to as CS) in the form of how to use word processors and database programs, which obviously is unappealing to many and gives a very incorrect perception of CS. Any *real* CS in high schools was very uncommon, as was programming. There were some Unit Standards students could do, but because of the type of standard(pass or fail rather than graded), many academically inclined students were not interested, and the subject was perceived as "easy" or for lazy students. To try and address this, some schools offered the graded standards, borrowed from the Technology area, but these adaptations were poorly done without an understanding of the content, and subsequently provided another incorrect perception of CS [18]. The reports stated that it was time for CS to be its own subject and to be taken seriously by high schools.

[1] `https://www.youtube.com/watch?v=v5yeq5u2RMI`

From this, a panel was formed. The Digital Technology Experts Panel was given the task of solving these issues of CS and Programming in schools. Later in 2008 and 2009, the panel released its recommendation. A new subject was to be created called "Digital Technologies" (DT), which encompasssed five areas — Electronics, Programming and Computer Science, Digital information, Digital media and Digital infrastructure[2]. Industry demand and enthusiasm pushed the rollout and very quickly, subjects were on offer to schools. In 2011, students were able to take the first stage of these subjects, with subsequent stages being offered in the years following.

These were offered under the NCEA (National Certificate of Educational Achievement) system, which is the most common assessment scheme for secondary schools in New Zealand. The system offers 3 levels. Level 1 is for students in Year 11 of School (their third to last year), Level 2 is for Year 12 and Level 3 is for Year 13. A unit of work in NCEA is known as a standard. A standard will provide credits and enough credits will earn you a certificate at that level. There are two types of standards. Unit Standards can be "Achieved" or "Not Achieved" and Achievement Standards are graded and can achieve "Merit" and "Excellence" as well. As explained above, more academically inclined students are attracted to Achievement Standards, as an NCEA level can be gained with graded endorsements of Merit and Excellence.

It was important that Digital Technologies attracted academically inclined students, so it is offered as a set of Achievement Standards. One of these at each level is for Computer Science, and students cover topics such as Algorithms [13], Human Computer Interaction, Encryption, Artificial Intelligence, Complexity and Tractability, Formal Languages, Computer Graphics and Vision, Software Engineering and Network Communication Protocols. It is worth noting that these are similar to their Maths and Science counterparts, and serve only as an introduction to what is available at tertiary level. The aim is an introduction to Computer Science that provides a positive and correct perception of the field. This project focusses on solely on AS91636 (also referred to as 3.44), "Demonstrate understanding of areas of computer science" which is a Level 3 standard.

Choosing to offer these standards in their high school is to be decided by that school, and a significant number did so in 2011, with over 2,700 students enrolling in the programming standards and 1,429 enrolling in the computer science concepts standards (1.44). These numbers increased in 2012 [3] and again in 2013 [6]. In 2013, there were 560 students enrolled in AS91636 (3.44), with a likely increase for 2014 and the future[3] because more teachers are being trained and more resources are becoming available.

While it was good to have a fast roll out and uptake of these standards, this left little time for the creation of student and teacher resources (examples, textbooks etc). Teachers were unable to predict what quality was required for student work. Universities began to take a role in providing professional development (PD) for teachers and ran such programs as Computer Science For High Schools (CS4HS)[4]. These programs have been welcomed by teachers, but not all teachers can attend and it is hard to reach those in rural areas.

For these teachers, and before programs like CS4HS started, an understandable lack of confidence

[2] http://www.nzqa.govt.nz/ncea/assessment/search.do?query=Digital+Technologies&view=all&level=01

[3] http://nceatechstats.com/digital/standards.php

[4] http://cosc.canterbury.ac.nz/cs4hs/

was present in teachers. It was found from a 2012 survey that only 44% of teachers felt confident teaching computer science [27]. As noted earlier, an unconfident teacher can lead to a negative impact on the students perception of CS. However, these teachers on average have 10 years of experience in the classroom. These classroom skills are crucial to presenting the content, however good PD and resources are still required for a successful delivery of content.

In addition to CS4HS, Outreach and other initiatives, the Computer Science Education Research Group in the Computer Science and Software Engineering Department at the University of Canterbury created the Computer Science Field Guide (CSFG) [2, 4]. This guide is an online textbook aimed at teaching the NCEA CS Achievement Standards. The book is split into chapters, each lining up with a topic from CS. The book has two versions. The student version has games, videos, tutorials and classroom activities to try out and the teacher version is annotated with detailed notes and tips on presenting and assessing content. The guide was released in 2013 and since then the book has been viewed by thousands of students and hundreds of teachers. The book has helped provide teachers with content and confidence for exposing students to a positive, accurate introduction to Computer Science.

1.2 Project Goal and Objectives

The goal of this project was to produce a chapter in the Computer Science Field Guide (CSFG) to assist in both the teaching and learning of the topic 'Network Communication Protocols'. The chapter contains interactive applications (interactives), classroom activities, tutorial text and an introductory video to teach the key concepts of the topic. It also covers the common problems and their solutions of networking protocols. The chapter will be aimed at final year students taking the DT standard AS91636 (3.44) of the National Certificate of Educational Achievement (NCEA) curriculum. Another important goal with this work is to make teachers feel supported and confident presenting the material, leading to a higher adoption of, and student enrollment in, AS91636 (3.44). We also expect to see higher grades. A special teacher version of the CSFG chapter is available on request) is also needed to guide the teachers through the topic and teaching it.

1.3 Report Outline

Chapter 2 examines the context of this work, what the requirements are for the standard, and existing work in this area. Chapter 3 looks at the possible content, depth of topics and the new material needed. Chapter 4 looks at the design and interaction of the classroom activities, interactive in-browser experiences (interactives), video and the tutorial text. Chapter 5 presents the evaluation of these components and Chapter 6 presents conclusions.

This research has covered the requirements for multiple components of a chapter, which are:

- written content
- video

- classroom games
- interactives/online games.

It is important to note that each component has its own background, method and evaluation. These will be discussed in each section below.

1.4 Output of Research

Overall, three papers have been published from this research. The material about the interactives' development and evaluation has been published in The 9th Workshop in Primary and Secondary Computing Education (WiPSCE '14) [19], which has an acceptance rate of less than 30%. Background work about the CSFG has been published in the International Olympiad in Informatics (IOI) Journal [5]. Results of the introduction of Computer Science in New Zealand High Schools, and the CSFG's impact on this, has been accepted and presented in New Zealand's Conference of IT (ITx) [6], specifically by the Computer Science Association of New Zealand (CSANZ).

2 Context of Work

This chapter details the NCEA Standard and previous work in the area of teaching networking protocols. While there is a lot of work in this field already, very little of it is suitable for high schools because it is aimed at tertiary education. The nature of the CSFG and NCEA provides some additional constraints in which to work in.

2.1 Requirements for NCEA Standard AS91636 (3.44)

This work is aimed at students and teachers sitting and teaching the NCEA Achievement Standard 91636 (3.44) in Level 3 of DT. In this standard, students have to choose two topics from Complexity and Tractability, Formal Languages, Graphics and Visual Computing, Intelligent Systems, Network Communication Protocols and Software Engineering.

Since each credit represents 10 hours of student time, and this standard is worth four credits, students can be expected to work up to 40 hours on their reports (including classroom time), which gives 20 hours for each of the two topics selected. This means that there are only a few hours of classroom time available for teaching Network Communication Protocols.

The students are assessed in the form of a report. This report is to be around 6 - 10 pages long, and is to cover two areas (network protocols being a possible choice for one of them). For the report, a student is awarded either an *Achieved, Merit, Excellence* or *Not Achieved*. Ways to achieve these levels are described in the table below. A report should use the "students voice" and be in their own words. Personalised examples are the best way to do this, where the report contains examples the student has generated personally, rather than examples taken from a textbook or website. One of our goals is to create content which lets students have a unique and personal experience so that their reports are also unique. An additional goal of this work is to see students achieve high grades for their work, by supporting both them and their teachers, by giving the students tricky questions to think about that require discussion and therefore lead to excellence answers. The criteria for grades is outlined in figure 2.1.

Achievement	Achievement with Merit	Achievement with Excellence
Demonstrating understanding of areas of computer science involves:	*Demonstrating in-depth understanding of areas of computer science* involves:	*Demonstrating comprehensive understanding of areas of computer science* involves:
• Describing key problems that are addressed in selected areas of computer science • Describing examples of practical applications of selected areas to demonstrate the use of key algorithms and/or techniques from the.se areas.	• Explaining how key algorithms or techniques are applied in selected areas • Explaining examples of practical applications of selected areas to demonstrate the use of key algorithms and/or techniques from these areas.	• Discussing examples of practical applications of selected areas to demonstrate the use of key algorithms and/or techniques from these areas • Evaluating the effectiveness of algorithms, techniques, or applications from selected areas.

Figure 2.1: Sample of the Level 3 Digital Technologies 91636 (3.44) Common Assessment Guide showing the marking schedule of the standard[2].

In DT in NCEA, there are also standards on digital infrastructure at all levels. We take care with Network Communication Protocols to not overlap or confuse students with this area. Focusses of these standards include "basic digital infrastructures[3], Local Area Networks including both conceptional[4] and implementational[5], and Wide Area Networks, similarily conceptual[6] and implementational[7]. There is some focus on addressing and routing in this area which we are careful to avoid.

2.2 Related Work

This research has four components, these are (1) written content, (2) video, (3) classroom games and (4) interactives/online games. It is important to note that each component has its own background, method and evaluation.

2.2.1 Existing Written Resources

A large portion of the existing work in this area comes from Cisco Systems [16] who have proprietary qualifications for secondary and tertiary students. Their content is taught in some high schools, especially with those who have a heavy focus on practical subjects such as Digital and Material Technologies (woodwork, metalwork etc). However, Cisco's qualifications are more focused on configuration of a network and the OSI model of the protocol stack. Computer Scientists prefer (according to interviews with academics, see below) to look at the Transmission Control Protocol over the Internet Protocol (TCP/IP) model of networking and to focus on the higher levels of these stacks (transport and application layers). The Cisco content would be more appropriate (and is used) for the infrastructure

[2] http://www.nzqa.govt.nz/nqfdocs/ncea-resource/schedules/2013/91636-scg-2013.doc
[3] http://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2014/as91080.pdf
[4] http://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2014/as91377.pdf
[5] http://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2014/as91378.pdf
[6] http://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2014/as91641.pdf
[7] http://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2014/as91642.pdf

standards mentioned above. Implementing the Cisco teaching systems in a school is also time consuming and would not be appropriate solely for AS91636 (3.44). For these reasons, we have chosen to leave this out of the chapter in the field guide.

Apart from from this, there is little research in this area about what to teach in high schools. Most research details plans or ideas for teaching tertiary level students. However, teaching secondary topics is a lot different, and this will be discussed in chapter 3.

2.2.2 Existing Video Resources

With the application of this research being to create a chapter in the CSFG, the resulting video must be similar to the others in the guide, a short 1-5 minute introduction to the content (which in itself is an introduction to the field). A chapter may also included other videos later on.

There are many types of videos online about Network Communication Protocols, most of which are on the popular video sharing website, YouTube[8]. Whilst there has been little research on what makes a good computer science education video, there are many attempts online, and these can be grouped into general categories.

The first of these would be the recorded lectures from universities around the world. These are good, but are usually of low recording quality, and the actual content itself is too advanced for an introduction, and a series of videos could last up to 10 hours (10 one hour lessons).

The second of these are those by Cisco and CompTIA — both of which offer proprietary certifications for their hardware. This content suffers from the same downsides as Cisco's written content: it is too specific and is too related to infrastructure.

Another type is the videos that try to attempt to explain a complex idea such as "the internet" in a limited time or simple language, such as "How the Internet Works in 5 Minutes"[9], "How Does the Internet Work"[10], "There and Back Again: A Packet's Tale[11] and Lee LeFever (Author of The Art of Explanation) has a video called "The World Wide Web in Plain English"[12] on the topic. Even NASA have made a video[13] for their Kids Science News Network.

A popular video (over 28,000 hits on YouTube) on this topic, and cited in academic works, is "Warriors of The Net"[14]. Warriors of The Net is a 13 minute long movie that follows an Internet Protocol (IP) Packet from a user's computer to a web server and back again. The video was produced in the early 2000s and is animated with a narration. The humour and graphics are outdated, the music is old-fashioned and some of the jokes might not be appropriate for high school students (such as a request to www.SEX.com being blocked by a firewall). The video also skips a lot of TCP and places emphasis

[8] http://www.youtube.com
[9] https://www.youtube.com/watch?v=7_LPdttKXPc
[10] https://www.youtube.com/watch?v=i5oe63pOhLI
[11] https://www.youtube.com/watch?v=WwyJGzZmBe8
[12] https://www.youtube.com/watch?v=rkOPWeuoW2M
[13] https://www.youtube.com/watch?v=UaPwZtrgvMQ
[14] https://www.youtube.com/watch?v=RhvKm0RdUY0

on various stages of a request/response lifecycle.

From reviewing the background videos, and keeping consistency in mind with the rest of the CSFG, it would seem a short, relatable, plain English video would be best. We define relatability as the ability for a student to recognise the problem or an analogous problem. We use these insights to guide the development of the video, discussed in chapter 4.

2.2.3 Existing Classroom Exercises

While teaching protocols in high schools is new, there is already a small selection of classroom games developed for this purpose.

Developed by Code.org, "The Internet"[15], is a game to demonstrate the reliability of various physical mediums. Students transmit messages while pretending to be one of three transmission methods, wireless internet (WiFi), DSL and fibre optic. The students who represent WiFi carry the message (a piece of paper) on their heads, DSL, on the back of their hand and fibre, held with two hands. Whilst this has nothing to do with protocols, it does demonstrate an interesting point and gets across the fact that the internet does have reliability issues. Hence the need for protocols in the first place.

These reliabilty issues come to the foreground in a classroom game for teaching a TCP-like protocol called "Tablets of Stone"[16]. It teaches key concepts constructively. It is part of the "Computer Science Inside" project, and is described in more detail by Cutts *et al.* [12]. In this activity, some students are chosen to be "governors" and the goal of the game is to send messages to each other via "tablets of stone" (usually pieces of paper). However, each message is to be delivered by a messenger, who, by selecting a random fate for each "tablet", may either *deliver* the message, *delay* the message, or *throw away* the message. The analogy to packet switching is clear, and so the goal is for students to be able to design protocols for sending messages using the tablets in a way that communication is still successful despite the random delays and losses.

2.2.4 Existing Digital Games

A 2013 survey of 41 games [16] showed that there were seven games in this area, none of which are suitable for AS91636 (3.44). There is very little research on making actual games suitable for this area, although insights can be gained by considering other approaches. Other approaches to teach protocols often include the use of simulators. Zengin and Sarjoughian present an approach for teaching the Open Shortest Path First (OSPF) protocol [28]. However, their tool and others like it are best suited for tertiary level education. They can be complex, use little or no abstraction and do not provide the sort of engagement or entertainment like a game or activity can. They also note that simulators for educational purposes are very limited. From this we can conclude that it would not be a good fit for the CSFG, which is aimed at students aged 15 to 18, although it could be valuable to suggest some simulators in the CSFG as additional exercises.

[15] `http://learn.code.org/s/1/level/102`

[16] `http://csi.dcs.gla.ac.uk/workshop-view.php?workshopID=4`

Similar to simulations is the use of packet tracing, presented by Matthews [21]. Matthews suggests a novel approach of carrying out packet traces using freely available software (such as windump, tcp dump or WireShark) and completing a structured set of exercises to teach students about network communication protocols. Matthews' results look promising. However, for our context this system lacks scaffolding and may be inaccessible for new students to the topic. We think it has a place as a extension type exercise in the chapter, but it seems very complex as an introductory resource.

A popular piece of software in German High Schools is FILIUS [15] (seen in Figure 2.2)[17]. The software allows for design and testing of networks from the lowest layers of protocols to the top with a browser. The software is to be installed on a user's machine and then users can design networks, including switches, computers and other hardware. They also set properties on this. Network code is able to be edited and then the simulation can begin. The idea of visualising a network is appealing and positive for students, especially in a high school context. However, the software is very complex and offers a suite of tools and ideas we try not to approach in New Zealand high schools. It also requires some programming which we are avoiding.

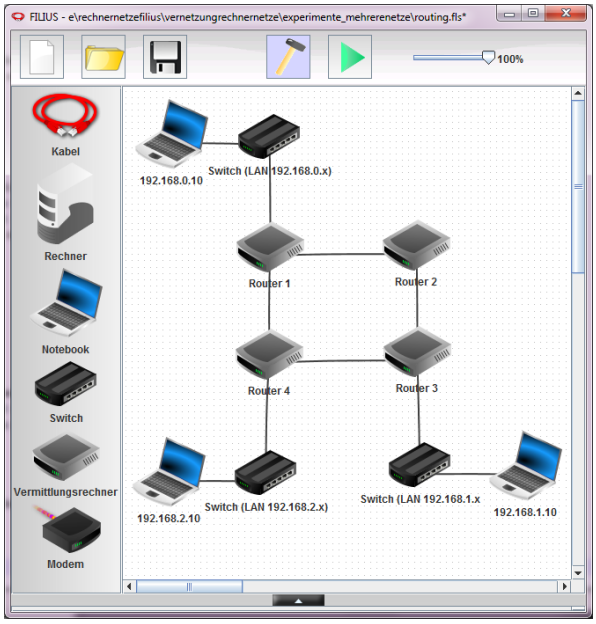


Figure 2.2: A screenshot of FILIUS, showing a layout of a network. Image from Informatik in der Schule

2.2.5 Other Approaches

Schwidrowski has made advances on teaching adults concepts of "internetworking" in the workplace [24, 25], derived from Freischlad's work on FILIUS above. The work also places importance on creating visualisations or abstractions and not focusing on implementation details. We are using a similar focus with our work. Schwidrowski defines strategies for designing learner's activities, but

[17] `http://inf-schule.de/kommunikation/rechnernetze/rechnernetzefilius/vernetzungrechnernetze/experimente_zweinetze`

is targeting more at adults whereas we are targeting students in secondary school. Adults tend to rely on their experience and learn differently. We are fortunate to be aiming at students who are so called 'digital natives', which allows for demonstrations and examples that they find familiar, such as referencing popular websites and services.

Another novel approach is presented by Richards [23]. His approach suggested fixing bugs in a networking library in order to become familiar with large, relatively well-kept code bases. The side effect that we are interested in is the learning of the protocol. The bugs become tools by which the students learn the inner workings of network protocols. However, after evaluating this approach, we do not find it suitable for students studying under NCEA as the code is more complex than would be expected for them to understand, and debugging large amounts of code assumes advanced skill. While the approach would be valid for advanced tertiary students, it is way beyond scope for what are essentially novice programmers.

Feaster *et al.* [14] introduce "Serious Toys II", which used a lightbulb and a series of software and hardware components to demonstrate each of the Internet Protocol layers: Transport, Network, Data Link and Physical. This is a promising concept, and the results of the paper are promising. However, the use of such a system is best suited to several weeks of teaching, where considerable interaction with the students is required (the paper mentions guiding students towards using certain protocols). Also, the approach assumes a level of programming that is too advanced for high school students. In addition to this, distributing hardware components could be a barrier to accessing the activity, and the CSFG focuses on digitally accessible demonstrations and interactions that can be used using only a web browser.

A common thread found amongst other prior work is the idea of combining protocols with other topics such as coding (information theory) or network security. These are also tightly linked with learning network configuration, such as the Cisco networking courses. Gramm et al. [17] described their classroom activities of using a mail server to teach security details. However they also described a system to teach protocols that included using string under a locked door and having students come up with a system of short and long pulls to communicate.

A related activity is Curzon's "Locked-in Syndrome" problem, where students need to work out how to communicate where one person cannot speak, but can only blink their eye [11]. Both of these challenges initially involve coming up with codes, but students soon learn that they also need a protocol to deal with errors (such as misinterpreting a string pull, or accidentally blinking when it was not intended). These are interesting ideas, and could provide a useful physical adjunct activity to the game we have developed. This has been included in the CSFG chapter.

Sloan and Schlindwein [26] describe a remotely accessible networking laboratory and set exercises for students to complete. The lab exists within their department and students access the equipment via a simple terminal computer system. This removes the need for students to configure their own equipment, saving time and costs. Their results are positive and the use of a hands-on component was popular among students. However, if we were to provide such a service as part of the field guide, we would encounter difficulties scaling to many classrooms. We also believe this approach is too advanced for high school level students. However, it might be well suited as part of the teaching of infrastructure standards in conjunction with a Cisco qualification provider.

From background reading, it is clear that several attempts have been made in this area. While all of them are novel, fun and suitable for their intended audiences, they do not necessarily apply well in a High School setting, or have a place in the CSFG. However, they provide guidance for the content covered and its delivery.

3 Choosing the Content

This chapter details the choosing of the content for the Network Protocols chapter of the Field Guide. It is important to reach the right level of depth and breadth to give a suitable *introduction* to the topic of Network Protocols.

3.1 Possible concepts and desired learning outcomes

Before any development could be done on the chapter, the key concepts needed to be decided. These are broken down by the standard and by the Common Assessment Guide for AS91636 (3.44)[1] into three areas: *Algorithms*, *Techniques* and *Applications*. These closely tie to AS91636 (3.44) and are used for marking and assessment as seen in 2.1.

To first guide our choice of concept, I consulted the existing key concepts provided by the Common Assessment Guide for 3.44. These included:

> **Algorithms**: (techniques are more relevant to this area than algorithms)
>
> **Techniques**: packet switching, handshaking, acknowledgement, authentication, checksums, wireless and wired security.
>
> **Applications**: many real protocols (eg TCP/IP) could rapidly become overwhelming at this level. Protocols that students could investigate that are less complex are DNS, UDP, HTTP (get and post), the addressing part of IP, SMTP and CDMA, and internet security protocols such as SSL, IPSec, and PGP.

To us, the ideas here seemed muddled, unfocused and too varied for a proper understanding of network protocols. I decided to further review and refine these ideas ready for teachers and students. I also hope to feedback these changes to NZQA.

3.2 Discussions with Academics

To initially refine the ideas and concepts from the Common Assessment Guide, I interviewed four academics from two universities. Each is involved in either teaching network protocols and/or has

[1] `http://www.nzqa.govt.nz/nqfdocs/ncea-resource/schedules/2013/91636-scg-2013.doc`

experience with the NCEA Achievement standards. I discuss the key points from these discussions below.

Academic #1 — *Victoria University of Wellington*

The first academic, who has published papers before on NCEA Computer Science, noted that teachers have, in the past, confused Network Communication Protocols with infrastructure. They emphasized that reliability was a big concept in the topic, and that the Application and Transport Layers should be approached. They also pointed out, in a previous year (AS 2.44) students may cover coding, encryption and compression, so this knowledge may be assumed. They also warned that Tablets of Stone may take up to an hour when played. Additional ideas included teaching routing and addressing and a coding project ,such as building a simple chat client on top of TCP.

Academic #2 — *University of Canterbury*

The second academic suggested that the most valuable comparison for students to make is that between the TCP and User Datagram Protocol (UDP). They also mentioned that the notion of addressing was important. The academic, a researcher in network security, also recommended covering topics from this area.

Academic #3 — *University of Canterbury*

Another academic suggested we take a step back and focus on what a protocol actually is, what protocols do, what can they achieve and what they give us. They said it is important to focus on the system as a whole, covering the need for layers (such as those in the OSI model/Internet Protocol suite). This could lead to discussion on how a layering system ensures effective and efficient communication. They also discussed transmission methods such as time division and frequency division multiplexing. They also recommended their research field, pseudorandom number generators.

Academic #4 — *University of Canterbury*

This academic had some interesting insights from teaching tertiary level Network Communication Protocols. Their first suggestion was the use of analogies and examples. The first was giving a postal system analogy, where each layer of the internet stack would roughly line up with how the postal system works. The second was what would happen if you had to send a whole novel one page at a time on the back of postcards would they all get there? What about the order? They also liked a boat on water vs a plane in the air metaphor as the bottom two layers (transport mechanism, medium).

They also confirmed the suspicion that the lower levels of the internet protocol stack can be avoided, not only with the infrastructure standards in mind, but also with the sheer complexity of the lower

network layers. They also mentioned that addressing could be covered.

Layering and encapsulation was also discussed, and there were three approaches to this. Top down, discussing the application layer first, then moving onto the transport layer. Bottom up, refers to the opposite direction of teaching. A spiral approach to teaching refers to giving an overview first, and then talking about each layer (order irrelevant).

They also stressed the fact that we should not dive into specific technologies too much, and keep the student mindful of where they are in the big picture. We need to answer the important question — can we actually guarantee data transmission?

These conversations with academics were a great help and guided the choice of concepts and inspired other components of this research.

3.3 Depth of Concepts

At the tertiary level, information is presented in full. There are no layers of abstraction, "half truths" (incomplete descriptions) or avoiding details. In secondary schools, we must use such techniques to avoid overwhelming students. We can present the information in a simplified, abstracted model, giving merely an introduction to the topic rather than a fully detailed course. We are also severely limited by time. While a standard networking course at university may last a few months of teaching time, at most this topic will be given a week in schools. The goal here is to give an introduction to the key concepts of the topic, to provide correct perceptions and to have students answer on what they have learnt, and show how much they have thought about this.

4 Design and Implementation

In this chapter we discuss the design and implementation of the four key parts of the CSFG chapter, the text, the video, the classroom games and the online game. The components each have their own design thinking but also inform the others to keep ideas and messages consistent.

4.1 The Tutorial

The chapter's main component is the tutorial section. The text contains information for both students and teachers. In this text, we take the student through an introduction to protocols, then talk about specific protocols. For this, we first needed to review some protocols for their pedagogical value and decide which ones to use in the chapter. These choices will inform the design of the other components of the chapter.

4.1.1 Pedagogical value of Internet Protocols

For each protocol, we decided to rank each in terms of several factors. These were:

- Complexity (either Low, Medium or High)
- Ability to personalise an experience (important for the report)
- Ability to demonstrate (in a school environment, where locked-down systems are common)
- Relatability of the problem the protocol solves.

At the application layer, we reviewed DNS, FTP, HTTP, IMAP, POP, RPC, SSH and IRC. We cover HTTP and IRC first as they were rated most favourably and were chosen for the chapter. The others are included for completeness.

HyperText Transfer Protocol

HTTP is a **medium** complexity protocol which deals with transferring all data when you visit a website. The ability to have a personal experience is trivial, with each student merely loading a website

of their choosing. The ability to demonstrate is also easy, with a user simply loading a web inspector, which all reasonably modern browsers include. The request/response mechanism of the protocol is also easy to create an analogy for, and a suitable one has to be chosen.

Internet Relay Chat (IRC)

IRC is a **medium** complexity protocol which allows two or more people to communicate. Put simply, it is a chat protocol. The ability to have a personal experience is also easy, as it involves having a conversation with another person via IRC. IRC clients and servers are available online through web applications, so are easy to access and no software needs to be installed. Chat programs are also nothing new for students of this generation, who are familiar with Facebook's chat system.

Domain Name System (DNS)

DNS is a **medium** complexity protocol which converts a domain name into a server address. The ability to have a personal experience with it is similarly easy to HTTP. However, the ability to demonstrate DNS is hard, and may require access to a DNS server. The idea of DNS is relatable, a simple telephone book (white pages) analogy would be suitable.

File Transfer Protocol (FTP)

FTP is a **medium** complexity protocol used for transferring files. FTP is common, but is difficult to demonstrate. One idea would be to set up an FTP server for use in the CSFG, but with a potentially global audience, security and ensuring appropriate use would be tricky. However, the idea of transferring files is an understandable concept.

Internet Message Access Protocol(IMAP) and Post Office Protocol (POP)

IMAP is more complex than POP, However the two would be presented together, as they both serve to send emails. This would allow students to compare and contrast the protocols and their features. Seeing the protocol in action is possible with some email clients, which can display header data on demand. The problem is also relatable because email is commonplace.

Remote Procedure Call (RPC)

RPC is a **high** complexity protocol which lets a client communicate with code on a server, and this makes it difficult for students to use. A likely use case would be students downloading, compiling and executing code. This is too much to expect from students at this age. However, explaining concepts of RPC looks to be easier than some other protocols.

Secure Shell (SSH)

SSH is a **medium** complexity protocol which allows a user to log into a remote machine and execute commands through a command line interface (CLI). The command line is often disabled in school networks. Also, using SSH in schools requires having a server to connect to, which presents the same difficulties as FTP. The concept of navigating a file system in a text based interface is also very unfamiliar to most high school level students these days.

At the transport layer, we evaluated the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). We decided to include both in the chapter.

TCP and UDP

Both TCP and UDP are **medium** complexity protocols which handle the transfer of packets. Both are tricky to demonstrate in a high school environment, but can be done with something like Wireshark[1] and is mentioned in the guide. However, the desired effect of Tablets of Stone and Packet Attack are to demonstrate similar systems, so we consider these a personal experience with TCP and UDP. Teaching these protocols is also achieved in the same way. Analogies and teaching examples can also be developed to make the problem relatable.

4.1.2 Teaching TCP, UDP, HTTP and IRC

We have chosen TCP, UDP, HTTP and IRC for teaching in this topic. These protocols come from the top two layers of the Internet Protocol Suite. With each protocol, we are guided by the NCEA marking schedule, focusing on key problems and their solutions.

HTTP

Our focus with the HTTP section was to make the student aware of the protocols' omnipresence. With the CSFG being an online resource, we can start the HTTP section of the chapter with these remarks:

"The HyperText Transfer Protocol (HTTP) is the most common protocol in use on the internet. The protocol's job is to transfer HyperText (such as HTML) from a server to your computer. It's doing that right now. You just loaded the Field Guide from the servers where it is hosted. Hit refresh and you'll see it in action".

This "right under their nose" approach is very powerful in a pedagogical sense as students feel connected and related to the material, and they can see it in use. For the HTTP section we identified the first problem the protocol solves is to transfer HTML, images, css. This is solved by HTTP's request/response system, so we focus on explaining this in the chapter.

[1] `https://www.wireshark.org/`

We chose to use a shop analogy. We use a conversation between the customer and shopkeeper to describe a request, the response (what the shopkeeper says) and resources (money, goods). We also use cartoons to display these to break up the large blocks of text shown in figure 4.1 and figure 4.2.

Figure 4.1: A customer buying a can of soda — analogous to a client doing an HTTP request.

Figure 4.2: A customer buying a can of soda with cash — analogous to a client doing an HTTP request with resources.

We then talk the second problem the protocol solves, and that is modifying content on a server. This is done by talking about the different types of request types (HTTP methods) and we use a sales rep/shop keeper analogy to describe restocking shelves using methods such as DELETE, GET, POST and UPDATE.

Lastly we discuss the HTTP problem of *failing gracefully* and the solution to this, error codes. We point out that pages can error with code 404 (a commonly seen error) and we even point out what a 418:Teapot error is[2].

We then encourage the student to open the web inspector and look at request response headers and information. Here they see these error codes, the response data, the request data and real life examples of the concepts above. In a university environment, this might be done with the command line with a tool such as cURL[3]. However, the web inspector is more visual, more likely to be accessible and more novice-friendly. We were careful in this section not to talk about other concepts of HTTP that relate more to web development or that are extraneous. For these reasons, we do not mention sessions,

[2] `http://www.google.com/teapot`
[3] `http://curl.haxx.se/`

idempotent requests and synchronous and asynchronous requests. We also do not talk about state vs stateless. In the teacher notes we describe how to obtain Google Chrome (which we found has the easiest to use Web Inspector), and just some information about the risks of using a Web Inspector.

IRC

Next in this section was IRC. This section was a lot simpler than HTTP. We encourage students to join or set up an IRC channel of their own on a system like Freenode[4] and start talking.

There are some aspects of the protocol we cover, such as the use of commands like /ADMIN and /DIE. Another is the concepts of channels, and naming conventions (typically a topic noun with a leading single or double hash). The aim is of this section is for students to try it out and explore, so the text content is kept deliberately light.

In the teacher notes we describe the risks of joining public channels and describe the process of setting up their own for their classes use.

TCP

The TCP section introduces one of the most important concepts in this topic, the concept of packets. We start by talking about why information needs to be split into packets. We then go on to talk about the key problems at this level, packet loss, corruption and delay. We talk about these problems and then let the student play *Packet Attack* [19], a game we have developed, described in detail below.

After the student plays the game, we talk about some of the solutions TCP uses. These include ordering, checksums and acknowledgements. Once again, the motivation is the standard and the guidance from NCEA. We are trying to show the problems and their solutions, and then suggest or let the user discover their effectiveness.

In the teacher notes we recommend answering on TCP (and UDP) for higher grades.We explain that TCP is better for reliable communication, but is slower and UDP is better for real-time applications. A teacher's section is also assigned to *Packet Attack*, describing level details and the analogies it makes.

UDP

The UDP section explains that the protocol does not guarantee against lost, duplicated or out of order packets, however packets do have checksum. What we stress in this section is that UDP is suitable for realtime applications such as *Skype*.

[4] `http://webchat.freenode.net/`

4.1.3 Projects

Each topic in the chapter has an associated project designed to engage students with the protocol, and try to answer some questions about it. For each project section, there are generous notes for teachers about what can be done for Achieved, Merit, Excellence in the report. Students are expected to use the project sections as a basis for their reports.

For HTTP, students are encouraged to use the Chrome Web Inspector to look at headers for requests and responses.

For IRC, students are encouraged to create an IRC channel, use the commands and see the responses they get.

For TCP and UDP, students are encouraged to play Tablets of Stone and Packet Attack. They can also create their own levels (see figure 4.12).

4.1.4 Teaching the Internet Protocol Suite

Academic #4 noted that there are three schools of thought for approaching the topic of layering; top down, bottom up and a "spiral" like approach. Whilst a spiral like approach is preferred, in the context of the CSFG, we carry out a top down approach. At the end of every CSFG chapter, a section called "The Big Picture" is presented. This section allows for high level information and related but not crucial topics. We saw this as a perfect place to introduce the concept of the Internet Protocol Suite.

We propose a situation that we are a website developer hoping to write an online music player. Obviously we do not want to install cables in every user's house, so something else needs to do that. These jobs are done of for us. This is when we introduce the idea of layer encapsulation and abstraction.

We introduce the Internet Protocol Suite and talk about how headers are added to a application data, then that forms a TCP packet, which then adds a header and becomes an IP Data, which adds a header, and so on as shown in figure 4.3. We discuss the notion of wrapping and unwrapping of packets. We compare it to a game of Pass the Parcel[5].

[5] `http://www.kidspot.co.nz/kids-activities-and-games/Party-games+7/Pass-the-parcel+67.htm`

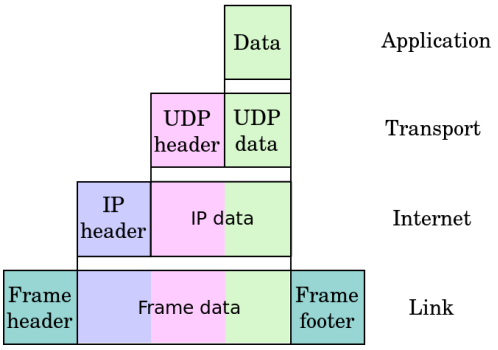


Figure 4.3: An image we use to show the wrapping of application data to frames.

We also present a simplified view of what a TCP segment looks like (see figure 4.4). We have chosen to keep only a few fields to maintain simplicity and avoid fields and flags we have not discussed in the chapter.

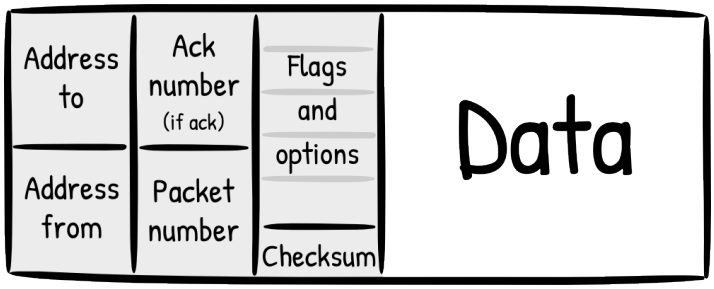


Figure 4.4: Our simplified packet, showing only some of the fields.

Another aspect of the chapter is the use of CSFG idioms such as the "Jargon Buster" (see Figure 4.5) and "Curiosity" (see Figure 4.6) boxes. These are used for when the jargon is getting a little heavy and for when there is something additional to mention that does not necessarily fit into the flow of the tutorial text.

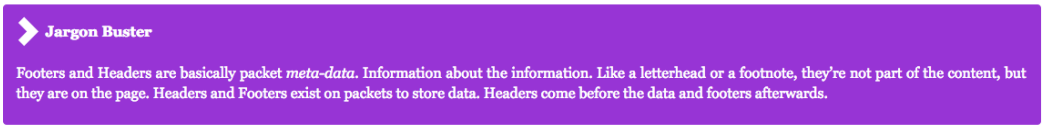
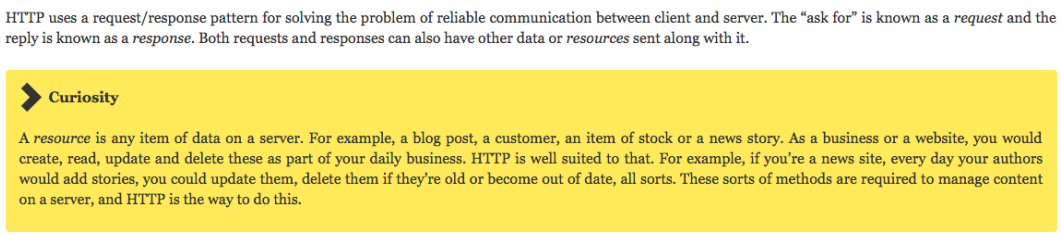


Figure 4.5: A Jargon Buster explaining Footers and Headers.


HTTP uses a request/response pattern for solving the problem of reliable communication between client and server. The "ask for" is known as a *request* and the reply is known as a *response*. Both requests and responses can also have other data or *resources* sent along with it.

Curiosity

A *resource* is any item of data on a server. For example, a blog post, a customer, an item of stock or a news story. As a business or a website, you would create, read, update and delete these as part of your daily business. HTTP is well suited to that. For example, if you're a news site, every day your authors would add stories, you could update them, delete them if they're old or become out of date, all sorts. These sorts of methods are required to manage content on a server, and HTTP is the way to do this.

Figure 4.6: A Curiosity inline in the CSFG chapter explaining what a Resource is in terms of HTTP.

We then talk about how TCP and UDP packets can change how long they are, asking the student to think of the consequences of this. This is a good leading question for excellence level responses in their reports. A small "further reading" section links to some YouTube videos on the topic.

A full copy of the chapter can be found online, along with the rest of the CSFG at `http://csfieldguide.org.nz/NetworkCommunicationProtocols.html`

4.2 Video

Each chapter in the CSFG begins with an introductory video. These videos have an aim of grabbing students' interest and give them an overview of the key concepts in the chapter. They are not intended to teach any of the content in the chapter, but by giving a 'big picture' view of the main topic, they give context to the lessons in the chapter.

4.2.1 Video Content

The next stage of the process was to decide what to include in the chapter. Academic #3 suggested at some point we cover what a protocol *actually is*. The aim of the video we created was to answer that. What is a protocol? Why do we have to use them? What can they do?

We aimed to answer these questions by making some key points.

- A protocol is a way of communicating formally.
- The internet is not reliable. Messages are lost, delayed and corrupted frequently.
- If we did not have protocols, the internet would be unusable.

These questions gave a framework on which to start thinking about possible examples on which to base the video. Some ideas that came up during this phase were Surgeons, Builders, the humour of misunderstanding, walkie-talkies, traffic lights, Police/emergency services radio communication, hands up in class.

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 4.7: Series of stills taken from the completed video

After discussing the video and the ideas with our production partner, Orange Studio[6]. We decided on some solid ideas we would like to have in the video:

- The narrator would be in multiple locations/situations
- Watching a packet/message travel around the world with an animation
- A airplane pilot theme
- Surgery, using the humour of failed communication
- Saying cut during the video to stop the video

These ideas were carefully balanced and included in the script and shots of the video. Some stills from the video are in figure 4.7.

[6] `http://orangestudio.co.nz/`

4.2.2 Video script

This section presents the full yet brief script of the video. Screen directions are in italics. The narrator in the video is Sam Jarman and the pilot is his father, Robert Jarman.

> Narrator: Talking is easy, right? We all do it everyday. Whether it's in person, by email or by text. But what happens when something goes wrong? What happens when what we're using to communicate just isn't re-rerrrrrr-reliable?
>
> *The video skips, pauses, appears to "glitch" .*
>
> Narrator: We use protocols. Protocols is a fancy word for a simple concept, communicating formally.
>
> Narrator: *(Dressed up in 1920s clothes, in a formal setting — see figure 4.7 (b))* The stock market is doing so poorly, whatever shall I do?
>
> Narrator: *Slaps forehead* Not like that!
>
> *The camera shows a light plane parked in the background*
>
> Narrator: Pilots use protocols.
>
> *The next shot shows the narrator in the plane, flying — see figure 4.7 (c)*
>
> Narrator: Their communication has to be reliable and efficient. They have to be able to communicate with other planes and various airports in order to stay safe. For example, how do they report their position? What happens if they don't understand what was said? What happens if they don't hear back? You can get into some pretty awkward situations.
>
> *A scene is presented between the pilot of the plane and the narrator — figure 4.7 (d)* Pilot: So what runway should we use? *The "tower" says, inaudibly runway one eight*
>
> Co-pilot: I think he said "Land on the cake". Pilot: Okay,we're landing on the cake.
>
> *The plane lands on a cartoon of a cake — see figure 4.7 (e)*
>
> Narrator: To avoid this, we use protocols. Protocols are a way to formalise communication such that we don't get ourselves into trouble. That's why pilots are always repeating stuff. *The video repeats always repeating stuff* and use special codewords to avoid being misheard.
>
> *A snippet of video from the pilot talking to an airport is played*
>
> Narrator: Pilots, emergency services, law enforcement, surgeons, builders and even you use protocols every day. They are essential to effective and efficient communication.
>
> *A scene showing the narrator as a surgeon is shown. The narrator asks for a scalpel yet receives a banana, and looks confused — see figure 4.7 (f)*
>
> Narrator back on the ground: A lot of people think the internet is bulletproof, but it's not. Messages are lost, corrupted and delivered out of order all the time. Think about when you send a message on *Facebook*. If we didn't have protocols, it would look something like this.
>
> *An animation of a message saying "Hey Anthony, how are you?" is shown to fly out of the narrators smartphone (figure 4.7 (g))around the globe (figure 4.7 (h)), getting bumped,*

*struck by lightening and slowly becomes "H3y *¿ $thony, How a$$re you?!?" (see figure 4.7 (i)). A confused Anthony stares at a computer on a desk in an office.*

Narrator: Luckily the message arrives on the other side of the world just fine. A protocol is in place to ensure each word or letter is delivered properly and if need be, they are re-sent.

Narrator: In this chapter, you'll learn about protocols such as TCP and UDP, which we use to build awesome applications on the web.

Narrator: Even people in the film industry use protocols. For example, to finish filming people say "Cut".

We hear the camera crew say "he said cut" and "okay, that's a take". The screen goes black and the narrator's voice trails off.

Narrator: All sorts of people use protocols, but in this chapter. Guys... guys? Are we still filming?

The video can be watched on YouTube at `https://www.youtube.com/watch?v=Hwqmu6pvr6g`. The video is provided on Vimeo as well as YouTube so that teachers are able to download it and play it in a classroom, as some schools limit access to YouTube, and it is available at `http://vimeo.com/106027170`.

4.3 Classroom Activities

For the classroom activities, I evaluated and include existing work in this area. Classroom activities have a simple purpose: to give the student a *personalised* experience with a key concept.

We started with Tablets of Stone. To test this game, we presented it to a group of seven computing teachers at a workshop, and had them trial the game.The first thing of note is the duration of the game. The game, from start to finish could take anywhere from 60 to 120 minutes, depending on the size and ability of the classroom. For many New Zealand high schools, this exceeds the maximum length of a class (usually 50-60 minutes). We also noticed that students might not be able to come up with ideas such as ordering or return/sending addresses on their own, so we found that stopping the game at key points for discussion is crucial to its success.

We have concluded that this game works best as a 10 minute demonstration of chaos (where parts of messages are lost), and then a longer discussion, with possible re-trials, of various solutions to this chaos. The time consuming of relying on students to solve the problems of packet switching over an unreliable network prompted us to consider inverting the game so that students were simulating the unreliability (attacking the packets) rather than trying to deliver them.

We wrote up these suggestions as part of a UC version of the game, and this is included as a file to download off the teachers version of the Network Protocols chapter of the CSFG. This is available as an appendix.

Students are encouraged to take photographs of their "tablets", and report about any unique situations

they experienced. This encourages the report to have the students' voice in it, rather than a rewording of an existing exemplar or resource. They can talk about what happens when they started using tablet numbers, ACKs and NACKs, timeouts, and what problems they solved.

The other game, which we have reviewed above, was created by the Code.org team. It is also included in the field guide as a possible activity. However, this game offers less value for a student, as it is hard to record a personal experience, and does not serve well for anything more than a demonstration of unreliability of the internet.

4.4 Online Interactive Game

Each chapter in the CSFG typically has one or more interactive game. For this chapter we developed one to help teach concepts in TCP and UDP. In this section, we describe the development of this interactive.

4.4.1 Design Decisions

We focussed our attention on creating a game for the chapter. Because the game is introductory for the topic, we selected a subset of problems and techniques that it would help students to engage with. The key areas selected were the important concepts that came up often in this context:

Problems: Packet loss, packet delay and packet damage

Techniques: Packet switching, acknowledgement (both positive and negative), check-sums, time-outs and ordering.

The design of the game was informed by earlier work on games for teaching CS [16]. Their discussion of the definition of educational games points out that it should be fun and attract users to engage, should have its own separate "world", have uncertainty so that there is not a prescribed outcome, have a set of rules and an element of fantasy or make-believe, and, of course, help the student to understand the topic better. They suggest that for a CS game to be engaging the designer needs to consider whether the game is active (the concepts are part of the game play), it should have "flow" (hard enough to be challenging, but easy enough to avoid frustration), and have longevity (it should be re-playable if the concept needs time on task to reinforce the ideas).

Based on our experience with the "Tablets of Stone" activity, students, unless given good guidance or scaffolding, can struggle to come up with solutions. A computer based game is likely to be played in such a way that external guidance will not be available, so we have chosen to invert the protocol design process by providing the solutions and having the student create the problem. This still allows students to think like the original designers of the protocol, identifying edge cases in which to test the solutions ahead of them.

For example, a timeout is a mechanism which re-sends a packet after a certain time period. It makes the transmission slower, so is there a real need for them then? Removing them will make the protocol more efficient, but as the student attacks the protocol they will find that there are cases where timeouts are needed.

This game play mechanics also allows for a more repeatable and uncertain game. If the game was to design a protocol to stop delays, deletions and corruption, then after just one or two plays, the value would be exhausted and users would know how to complete the game. This goes against the longevity that an engaging game should have [16].

4.4.2 Gameplay

The idea of this game is to present levels, of increasing difficulty, that slowly build towards a protocol like TCP.

Creatures representing packets (the blue objects shown in Figure 4.8) move across the screen from left to right, and as they cross the yellow striped danger zone in the middle the player can launch attacks on them to prevent them getting through. Each creature carries one letter from a message ("HELLOTHERE" is the complete message in Figure 4.8). At each level (except for the last) of the game it is the user's goal to *stop* the message being transferred accurately. Level one has no protocol to recover from lost packets, so it is trivial for a user to succeed at the level by simply causing any one packet to be affected.

The threats that the user can apply to the packets are delay (the packet arrives late), corruption (the content of the packet is changed), and deletion (packet loss). The choice the user has here makes the game uncertain and repeatable, and as the levels increase, protocols are introduced that are able to recover from the various attacks, making the player's job harder. The new rules (improvements in the protocol) are given to the player before each new level, so they can devise which attacks will be most effective. The number of attacks that the player can make are limited; there will be enough to stop the protocol working if that is possible, but the player will need to be strategic on the use of attacks for higher levels.

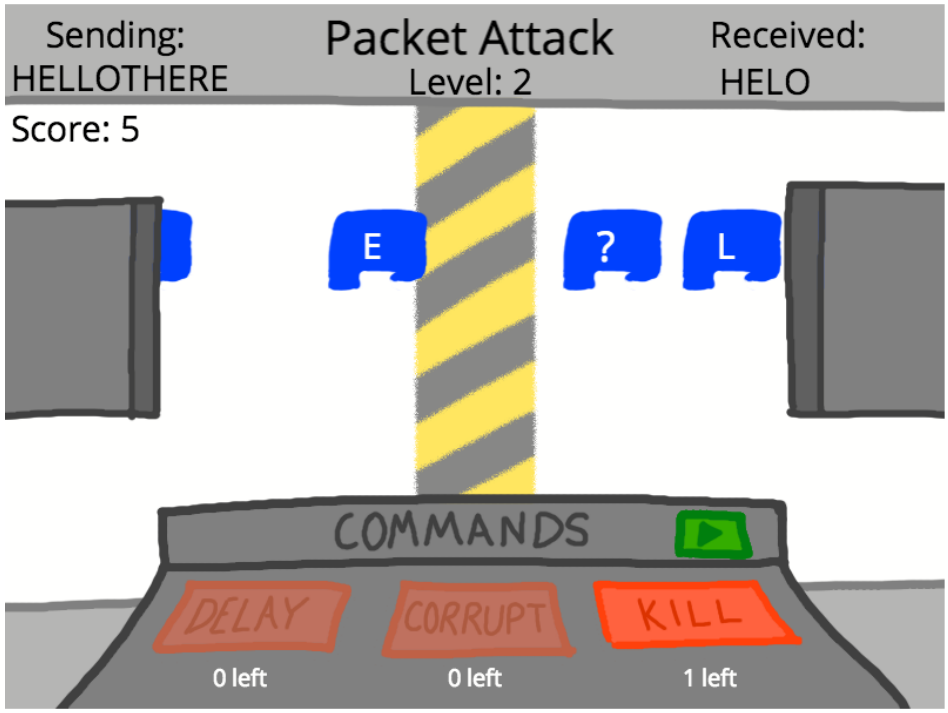


Figure 4.8: Screenshot of game showing controls, packet creatures with letters, effect of delays and effect of corruption

As the player goes up levels in the game, the protocol introduces new defences. Figure 4.9 shows the full range of defences, including sequence numbers, acknowledgements (acks) and negative acks (nacks) caused by timeouts. There are also checksums available as a defense against packet corruption.

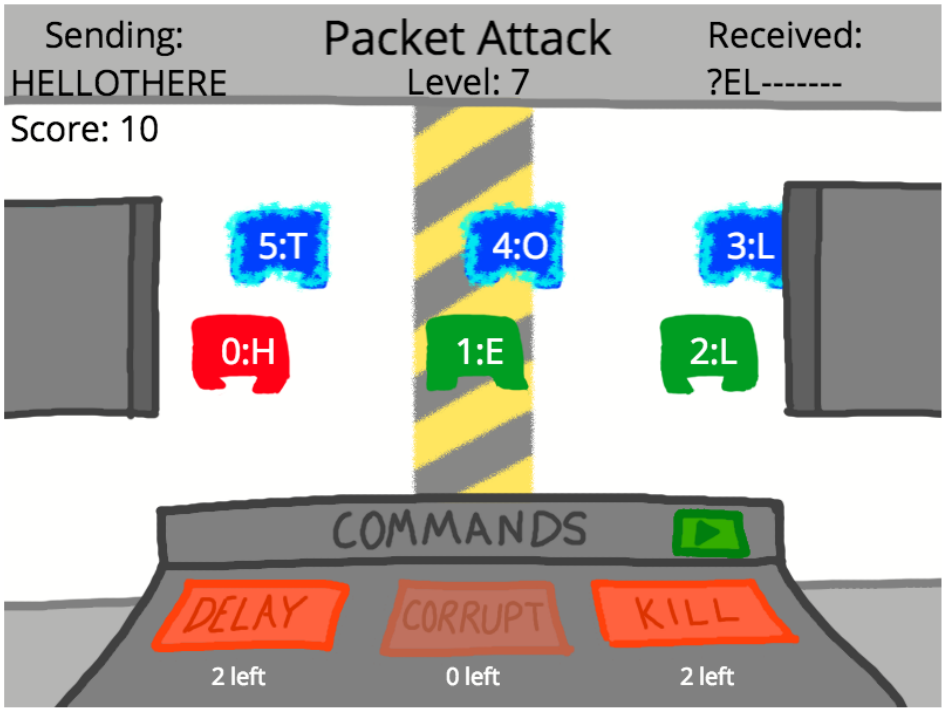


Figure 4.9: Screenshot of game showing acks (green) and nacks (red), numbered creatures, as well as incomplete data in the received message section

We chose not to allow deletion of acks or nacks in this early version of Packet Attack. While this moves away from the reality of network communication, allowing it adds complexity and may confuse users.

There was a lot of thought put into the both the names of the attacks and the representation of packets. It would have been ideal to find a metaphor that covered all of the elements of the protocols, but the complexity and reliability of network protocols do not map neatly onto the physical situations.

One metaphor considered was birds carrying the packets, reminiscent of RFC 1149, "Transmission of IP Datagrams on Avian Carriers"[7]. We note that RFC 1149 only offers "best effort" delivery, so it not able to recover from attacks on the birds. With the bird metaphor which we explored, delays were called 'Stun' (a poison that slowed the birds and caused them to lose their ordering), corruption/bit-flipping was called 'Zap' and deletions were called 'Confuse'. We did not use the term 'Kill' because we wanted to avoid violent terminology when it was not necessary.

This level of metaphor removed all computer science terms and presented a fun game simply based on the concepts that had a meaning in the physical world. The use of metaphor makes the game inherently make-believe, a property typical of engaging games [16].

However, during early stage testing on colleagues, they thought the pedagogical value would be lost in creating an a game just based on the metaphors, so we reduced the level of metaphors and re-

[7] `http://tools.ietf.org/html/rfc1149`

designed slightly. Our current iteration has "packet creatures" as packets, and three new verbs: 'Delay', 'Corrupt' and 'Kill'. The 'Kill' has returned since the packet creatures are fictional, and lack any resemblance to real life animals. These choices are purely superficial in terms of implementation — strings and images — however they hold a lot of value pedagogically, and we will continue to refine the metaphor based on feedback.

We also had to decide on a metaphor to represent error detection and correction (e.g. checksums). These need to suggest corruption of a packet can be prevented. We decided to use a shield analogy: the animated creatures have a fuzzy shield around them that is lost when the 'corrupt' attack is used (error correction is used, but cannot survive further corruption).

In Figure 4.10 the packet containing "E" has had its shield destroyed, and is now vulnerable to corruption. When the information represented on the creature is "corrupted" it turns into a '?'. Another possibility would have been to substitute a different character, but this could make it less obvious that the message has been assembled incorrectly at the end.

Figure 4.10: Packet creatures with and without their shields

As the packet creatures move across the screen, a user has a limited time to attack them. If the speed is too high the user may not have enough time and become frustrated, and if it is too slow, a player may become bored and lose interest. With some experimentation, we chose a speed that was in between these, adjusting it slightly during iterations.

The game also has a pause and resume button. The reasons for these are twofold. The first is to allow the student to take a break from the game and reflect on how to win. The second is to support assessment of student learning for the educational criteria that the game is supporting. For the particular assessment of this content in the NCEA standard, a student is expected to write a report that reflects on a personal understanding of protocols. We expect to see screen shots of the paused game with students discussing particular situations they create.

Because each student can experience different games, the screen shots and student discussion will be different in each case, providing the evidence that it is the student's personal experience. We are also considering implementing a rewind function and more control over the levels to make this easier.

When choosing a name for the game, we wanted to capture the concept that we are causing issues with segments of information. Initial ideas based on the bird metaphor were "Bird Strike", "Poultry Attack", "Messenger Birds" or "Bird Shoot". However, when we moved towards a packet creatures analogy we went back to the original code name of *Packet Attack*. We believe this conveys the actions the user will take during the game well, and provides a metaphor of what happens in digital networks.

4.4.3 Implementation framework

To choose a framework to develop the game in, we had several criteria we needed to meet. Firstly the framework had to be Free and Open Source Software (FOSS), as is the CSFG. It would need to run in-browser, as all the Field Guide activities are intended to be used without installation. It needed to be fast, and easy to develop with so we could quickly respond to feedback and iterate on game design. Lastly there were considerations about rural schools with low bandwidth, and use of old computers, browsers and operating systems in some schools. Currently the two main web technologies for displaying something like a game within a web browser are HTML Canvas and WebGL, both of which let JavaScript code render a scene. HTML Canvas was available in older browsers, with WebGL being available in newer browsers. Ideally a framework would need to support both, preferably with automatic fallback.

The website HTML5GameEngine[8] provided valuable information to make this decision, and using this we found the Phaser framework[9] which matched our criteria. Phaser gives us a standard set of features for a framework, which include a canvas, preloader, physics, sprites, animations, camera control, input, device scaling and mobile support. We use a subset of these features in the game.

After choosing Phaser as a framework, we tested performance by developing a small sample game with large number (1,000) of objects being tracked and updated every frame. From this we evaluated what operating systems and browsers the game can run on successfully, with a suitable frame rate (at least 20 frames per second).

The oldest operating system we tested was Windows XP, using various versions of Chrome, Firefox and Internet Explorer (IE). The latest versions of Chrome and Firefox both load the game and run at an average of 22 and 29 frames per second respectively. Unfortunately the latest version of IE on XP (8) does not load the game. In fact, it is only until you get to IE 10 on Windows 7 that it even loads, and even then the frame rate has an average of 16.45. IE 11 on Windows 7 performs well with an average of 30 frames per second.

To see the impact of this, we looked at the visitor data to the Computer Science Field Guide to see what proportion of readers would be affected. A total of 7.2% of users visit the guide using Windows XP. Of that 7.2% of Windows XP users, 17.51% of them use IE. This means, at most, 1.26% of readers would not be able to load the game using the Phaser framework.

To overcome this problem, the guide would simply recommend changing browsers to Chrome or Firefox, and if this was not an option (not uncommon for locked down school networks), to simply try the game at home.

On the most recent versions of the major three browsers, the game performs well at a high frame rate, some up to 60 frames per second.

Packet Attack is implemented in pure JavaScript using the Phaser framework. The main program defines levels which more code then interprets and produces levels out of. There are many parameters

[8] `http://html5gameengine.com/`
[9] `http://phaser.io/`

to a level, all of which can be permuted to make different levels. We have initially produced seven levels:

Level One: No defences. One packet. Students can beat by using corrupt or kill.

Level Two: Multiple Packets. 10 packets. Students can't stop all packets, but can corrupt and kill and delay for points.

Level Three: Shields enabled. 10 packets. Students can beat by using kill, but corrupt won't work.

Level Four: Numbering enabled. 10 packets. Students can beat by using kill, corrupt, but delay won't work.

Level Five: Numbers and shields. 10 packets.

Level Six: Numbers and acknowledgments. Packets will be sent back and resent.

Level Seven: Numbers, shields, timeouts and acknowledgments. Packets will be sent back and resent. This level is not beatable.

Figure 4.11 shows the introduction to level seven, showing a list of all the defences.

Level creation is the area that feedback from teachers will help shape the most. For this initial version, we tried to create a minimum set of levels to try expose students to what problems are solved by what solution. The intention is to help students pair the problem and solution in their understanding of protocols, as this understanding is crucial to succeeding in the standard. Further levels can easily be added, with the main one under consideration being adding the ability to attack acks/nacks.

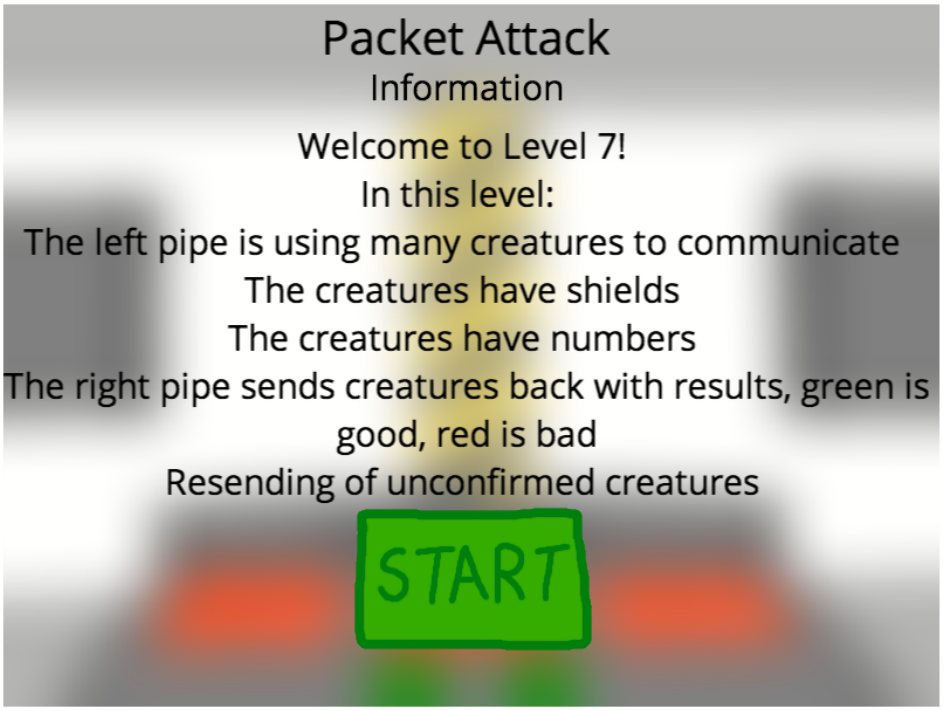


Figure 4.11: Screenshot of level seven listing all the defences.

We also created a fully operational level builder for Packet Attack (see figure 4.12), which is embedded in the protocols chapter of the CSFG. This allows for both teachers and students to create custom levels to help with both teaching and their reports on the topic.

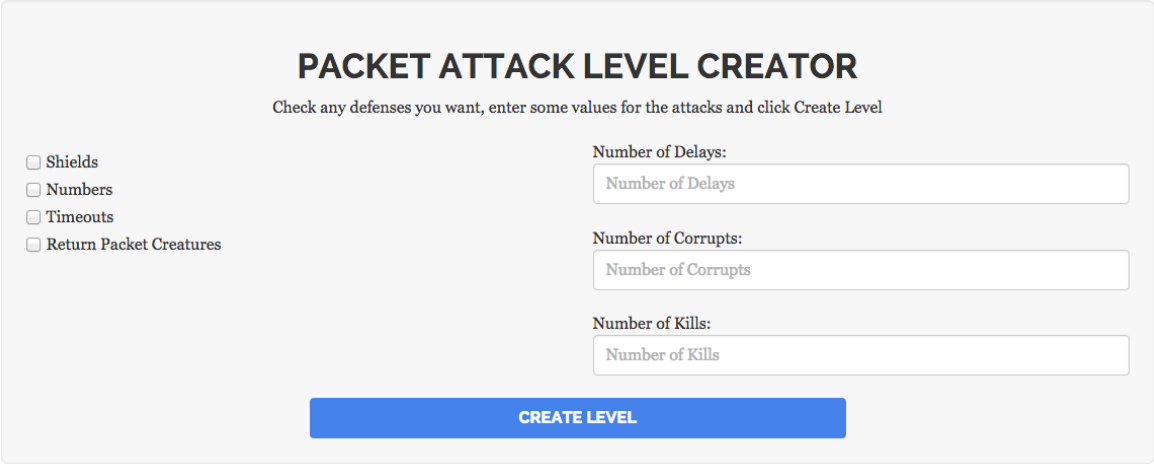


Figure 4.12: Packet Attack level builder — found in the Protocols chapter of the CSFG.

The final game is avaliable at `http://bit.ly/PacketAttack`.

5 Evaluation of Chapter

In this chapter of the report, we discuss the evaluations and results of the work this far. Ultimately, we cannot know if the work is successful until NCEA results come out in 2016, after the content has been taught in 2015. We expect to see students choosing Network Protocols as a topic for their report and through this work, achieving high grades for it. However, in the meantime, each individual component can be evaluated for its pedagogical effectiveness. We also conducted a pilot study of the completed chapter at a local high school, where an advisor helping with the class lead his class of six students (one female, five male) to get crucial feedback on the content. The study was largely a success; details are given below.

5.1 Classroom Activities

As mentioned previously, we trialed the original Tablets of Stone with a group of six digital technology teachers at a workshop. We noted two key observations. The first of these is the length of the game. To play the game in full, allowing undirected students to solve issues, implement new protocols and test them, it could easily last 60 to 120 minutes. The second is we noticed that students may not come up with ideas such as ordering or return tablets on their own. To address this, we implemented a revised version of Tablets of Stone which we have published alongside the chapter in the CSFG[1]. The version of the instructions clearly spells out the preparation and running of the game, and also suggests the game is stopped at key points, problems and their solutions discussed. We hypothesise this will help students form the mental pairing of problem and solution which is crucial for discussion in the report.

At the high school, the class played Tablets of Stone. The students used our improved version of the game, and the teacher prepared the "tablets" (small pieces of paper, see figure 5.1) ahead of time. We noticed the game ran much more smoothly than the original version we played with the teachers. However, we noticed that the game needed lots of tablets (the students ran out), which is something we have now added to the teachers instructions in the Tablets of Stone instructions. Of course, there is a lesson here about the the reliability of a protocol to be taught. We observed students coming up with the idea of sending an acknowledgement tablet and finding the solutions of numbering and timeouts intuitive. This was excellent to see. The students level of enjoyment was clear and when asked they responded very positively.

[1] `http://www.cosc.canterbury.ac.nz/csfieldguide/dev/dev/_static/files/UCTabletsofStone.pdf`

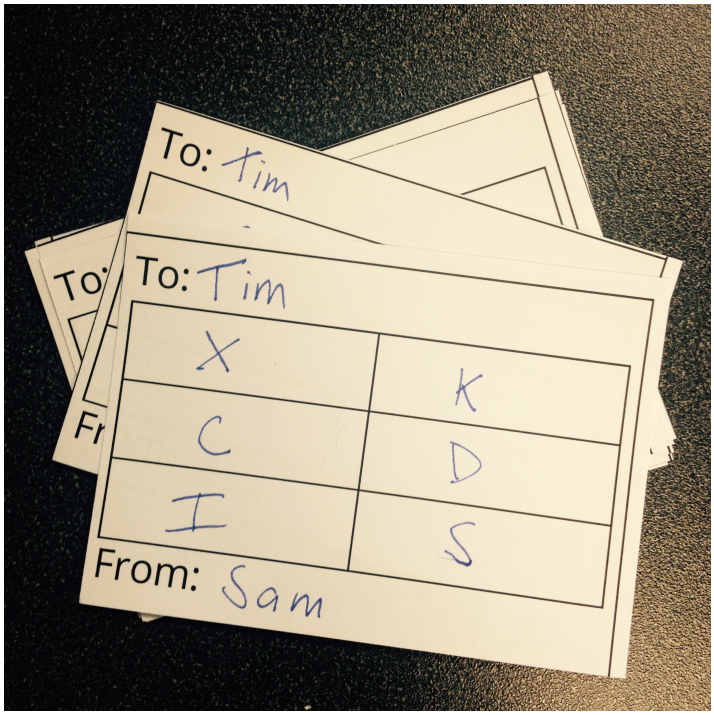


Figure 5.1: An example of a 'tablet of stone' students send each other other in Tablets of Stone.

5.2 Online Interactive Game

We can measure the likely value of the game in a variety of ways. An important concern in an educational setting is the *performance* of the software; often school computer labs will contain older computers, or students may be using a portable device (BYOD) that has limited computing power. Assuming that performance is satisfactory, the key concern is then the *pedagogical value* of the game — if it is applicable in classrooms, and covers the topics suitable. In terms of the guidelines for educational games [16], we also need to reflect on how active the game is, if flow is possible, and its longevity. The active nature of the game is built into the design, but we measure the flow and longevity based on live usage of the game.

Each of these are considered below for the first version of the Packet Attack game.

5.2.1 Performance of Game

The game uses HTML5 so that it can be used on a wide variety of browsers. We had tested the performance of the Phaser library by setting up a game that required various levels of computing power (up to 1000 sprites), which is more than required for the Packet Attack, which typically has up to 30 sprites at any one time. The 1000-sprite game was able to render at 30 frames per second even on a five-year-old machine running Windows XP. The final implementation of Packet Attack performs very well on desktop browser and even mobile devices. The low number of objects and

simplistic physics model means it makes very few calculations. Frame rates of 60 frames per second (the maximum) are typical on Intel Core i5 processors. Frame rates on mobile devices approached 30 frames per second, and based on estimations from the performance test, we predict up to 30 frames per second on older browsers and computers.

5.2.2 Pedagogical Value

A survey of teachers done by Mesaros and Diethelm [22] suggested that they prefer to use visualisations. The game has been designed essentially as a visualisation of a simulation of a protocol, but by introducing the interaction from the user they are engaged with the attacks and defences that the protocols are dealing with.

When the game was first released, it was released to the teachers who were members of the national association of computing teachers (NZACDITT) who would be using the CSFG.

Discussions from these teachers and their students provided positive feedback, with the exception of one issue: the game was released as a stand-alone game, with the intention of being embedded inside the completed chapter of the Field Guide, so some teachers found the lack of background information to be a hindrance on the game's pedagogical value for them. However, when provided with even just a small sample of background information they found the game was much more engaging, and that the concepts displayed were clear.

Now that the CSFG chapter has been released with the game embedded, these issues have subsided. Our best opportunity to see how students interact with Packet Attack was at a local high school. All six students played the game, and finished the levels within the expected 5 minutes. They spent some time discussing the game, and what they saw. It is clear the game promotes positive and pedagogically valuable discussion between peers.

They also got to try out the level creator to create custom levels. Students saw the advantage of this straight away, making levels with thousands of attacks and minimal defences. The boys preferred to try the damage the message the packet creatures was transferring as much as they could, while the female student spent time trying to find and point out edge cases. On the second day of the trial, students moved to writing their reports so were taking screenshots of the game and their custom using the pause feature we've built. These features were well received and lead to higher quality reports.

5.2.3 Engagement as a Game

The criteria of flow and longevity for the game were evaluated by using analytics from the use of the game online. We captured statistics of use via Flurry[2] using inbuilt analytics. In particular, we tracked the usage of the pause button, and what levels were started, passed and/or failed. We can also measure time spent in the game.

[2] `http://www.flurry.com/`

Users had a high success rate in the game — that is, most managed to prevent the delivery of the message and pass all the levels.

From the analytics we found that 53% of users succeeded at level one and moved to level two in their first attempt. We believe this low rate is due to learning the mechanics of the game, and 50% of first time failing users pass in their second attempt. The pass rates for the rest of the levels are high (70%-100%). We believe learning happens from both level fail and level success, and most of all with a combination of both. A formal evaluation will confirm this, but currently our focus is to be able to adjust the game so that we can achieve a pedagogically effective number of failures.

The median session length was 2.7 minutes, with most users spending 2 to 6 minutes in the game. This is to be expected for an educational game, and certainly aligns with our expectation of engagement length. Level repeats add longevity to the game. From the small number of sessions we have currently measured (246), we can see the expected decline in fail rate for each level as players repeat the level (more people pass the level on their second, third or fourth attempts).

The use of the play and pause button is high also, with 80% using at least one of these features. We also know that there have been 56 custom levels created and played with the latest version of the game over a period of 4 months.

What we can conclude from these statistics is the game is working and engaging as hoped. The design decisions have been validated and the features added have been used. The real value of this game is dependent on how much the student engages and how much the student understands what they are doing. This is what we are seeing from the analytics and trials.

5.3 Video

Validating the video's effectiveness quantitatively is challenging. If we look at the services we hosted the video on, we can gather some metrics. The video is hosted on YouTube and Vimeo.

As at October 17, 2014, the video is showing 606 views on YouTube. Of these views, there are 12 "thumbs up" and 0 "thumbs down". On Vimeo, a source which is less well advertised, there are 25 plays and 34 pageviews. We are yet to get any downloads, which is likely due to teachers with limited access to YouTube (rural area and/or blocked) creating lesson plans yet and teachers in urban schools being able to access YouTube directly. YouTube provides an unique second by second analysis of retention, so a user can view when users stop watching their video [1]. For our video, a graph is presented in figure 5.2. This is a good retention rate and looks like drop off is organic — that is to say no part of the video is particularly disengaging or unappealing. The average view duration is 64% which is 1:40 of the video. YouTube reports that most watchers stop watching in the first 15 seconds [1], so this is excellent retention, given how much of the general public have seen this video.

Figure 5.2: Retention section of YouTube's analytics area. Shows good retention. The y-axis is percent viewers and the x-axis is the video timestamp.

Unfortunately the video was not prepared in time for the pilot study at the local high school due to delays with the production studio, so there is no video feedback from the pilot study.

5.4 The Tutorial

Students at the high school, with direction of their teacher, looked at the TCP and UDP sections only. The students were engaged with the content and were able to take advantage of the fact the game was embedded for use.

In terms of metrics provided by Google Analytics, the teacher version of the chapter has had 80 pageviews, with the average duration on the page being 3 minutes 50 seconds. The student version has had 146 pageviews, with the average duration being 1 minute 14 seconds. The site average is 2 minutes 37 seconds. This is positive and indicates that retention is held, since users typically "bounce" off webpages within a few seconds, according to an academic we interviewed in the Human Computer Interaction field.

5.5 Teacher and Student Surveys

We also conducted surveys and email interviews with teachers and students to gain additional feedback for the chapter. These surveys were reviewed and approved by the University of Canterbury Human Ethics Committee. We recruited these participants via the NZACDITT mailing list. We had four teachers respond and were interested in looking at the new content and possibly teaching it to their class. We sent out the survey and sent reminders every two weeks. Even though it is late in the school year, and most teachers have moved onto other topics in DT, the teacher survey had four responses and the student survey had one response. Each survey had a series of question with a Likert scale response system with possible options being *Strongly Disagree, Disagree, Neither Agree nor Disagree, Agree, Strongly Agree* and *Not Applicable*, followed by an opportunity to comment on positive and negative aspects of the chapter.

5.5.1 Teacher Survey Responses

We asked teachers about several components of the chapter, and the effect they had on teaching the content in a short survey. Their answers are recorded in table 5.1.

Question/Response	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Not Applicable
The guide supported me to help students write a report for this standard	0	0	0	2	2	0
The video was a suitable introduction for my level 3 students	0	1	0	1	2	0
The video made sense to me and my students	0	0	0	1	2	1
My students enjoyed the Packet Attack game	0	0	0	1	2	1
My students learned from the Packet Attack game	0	0	0	1	2	1
I created custom levels of the Packet Attack game for my students	0	0	1	2	0	1
It was useful to be able to create customlevels of Packet Attack	0	0	1	2	0	1
My students understood the text	0	0	1	1	1	1
The guide supported me in teaching the content	0	0	0	0	4	0

Table 5.1: Teachers' responses to survey.

Please comment on anything in the chapter that was particularly helpful

- *It was great to have some practical applications for the students to experience and learn about Protocols. When I had first looked at this section, without the guide, I had found it difficult as all the information I was finding was going into a lot of depth and I was easy to go off on tangents and lose focus on the communication protocols. The chapter has made it easier to focus on the key elements and identify what the key problems, algorithms and practical applications are for Level 3. It also allows the students to create a personalised report through the packet attack creator.*
- *Great explanation! I haven't taught this this year but intend to do so next year to further expand the 'knowledge base' and see which ones the kids respond best to. I found the explanations in relation to the packet attack game both interesting and informative. Would an 'interest area' on say DOS attacks or virus actions be worth including??*
- *The packet game*

Please comment on anything in the chapter that was not helpful

- *All good!! keep up the fantastic support. It IS appreciated.*
- *No simple description of IRC. Could have tied in to Level 2 concepts encoding. Students are still using the resource so will have more info later.*
- *Seemed like TCP and UDP were more interesting and better explained. There were no suitable report questions for students in HTTP so skipped this, as students can work at their own pace through TCP/UDP report.*
- *Apart from creating a personalised report by using the packet attack creator, I don't see the point of the students having to create their own level - am I missing something?*

Overall the responses from the teachers are positive. The unanimous agreement that the guide supported them in teaching the content is excellent to note, as this was one of our big goals with this work.

5.5.2 Student Survey Responses

We also asked students about several components of the chapter in a short survey. Their answers are recorded in table 5.2.

Question/Response	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Not Applicable
The video was entertaining	0	0	0	0	1	0
The video made sense and was an interesting introduction to protocols	0	0	0	0	1	0
The Packet Attack game was fun to play and engaging for me	0	0	0	0	1	0
The Packet Attack game helped me understand the transmission control protocol (TCP)	0	0	0	0	1	0
The text was clear and easy to follow	0	0	0	0	1	0

Table 5.2: Students' responses to survey.

Please comment on anything in the chapter that was particularly helpful
All of it
Please comment on anything in the chapter that was not helpful
(No responses)

While the student survey does not provide much insight, the students at the school appeared to enjoy the content and were engaged with it during the lesson. They were asking the "right questions" (understanding what the protocol does) of their teacher, and the teacher reported feeling confident answering these (and he did so correctly).

5.5.3 Other feedback

We had other feedback from teachers who were very impressed by the chapter and found it a great new resource. A teacher reported their students found Tablets of Stone particularly useful and noted their students came up with some of the solutions but not all of them, as we predicted. Because of the revised instructions we prepared, the teacher was able to help the students through the game. The teacher reported that playing Packet Attack after Tablets of Stone consolidated learning. They pointed out the HTTP and IRC sections were not clear on what could be done for the report in terms of problems and solutions, and obtaining higher grades for these. Since then, we have added more teacher notes to this section with the guidance of a colleague who is an expert on the standards, marking student work, and an author of many sections of the CSFG.

Another teacher did not feel confident teaching the IRC content due to a lack of instructions in this area. We note this and in the future will look at adding more. However, they were impressed with Packet Attack and found it favourable over Cisco tools.

The level of feedback and richness of analytics will increase as the chapter enters mainstream use by students and teachers in 2015.

6 Summary and Conclusions

After a highly critical report from two organisations in 2008, the Ministry of Education created and released new NCEA standards for Computer Science in 2011. These standards are part of Digital Technologies and are aimed at providing a correct perception of what computer science is, and what problems computer scientists solve. A swift introduction of these standards meant that teachers struggled to deliver the completely new content effectively or with confidence. Subsequently, students' work and perception of the difficulty of CS suffered.

To assist schools and teachers with rolling out these new standards, the University of Canterbury Computer Science and Software Engineering Department's Computer Science Education Research Group started the Computer Science Field Guide (CSFG) [2]. The online textbook, now complete, provides information for teachers and students on each of the Computer Science subjects in the NCEA standards. The content solves many of the reasons students are not choosing ICT as a career field, by exposing them to CS in a positive and accurate way.

The goal of this project was to research and produce content for the Network Communication Protocols chapter of the CSFG. The chapter is intended to assist students with the Network Communication Protocols section of AS91636 (3.44), a Level 3 Computer Science standard. For this chapter we looked at previous submissions, prior work and attempts by Cisco in this area. To finalise content, we interviewed academics to explore key concepts, evaluated protocols from the Internet Protocol Suite for teaching ease and pedagogical value and we studied previous attempts. For the chapter we researched and developed four key components.

- We looked at classroom games and experimented and trialed these, adding our own set of improvements to and implementing a new version of Tablets of Stone.
- We developed Packet Attack, an original game for teaching the fundamentals of TCP and UDP.
- We scripted and shot a short introductory video on the subject.
- We wrote a tutorial featuring HTTP, IRC, TCP and UDP, as well as an overview of the Internet Protocol Suite.

From the early feedback from teachers, a pilot study at a local high school, and the examination of available quantitative data and metrics from analytics of the game, video and chapter, we are confident this content will be enjoyable, engaging and informative for students studying Computer Science at Level 3. The evaluation gives confidence that the content will help students to write reports on

Network Communication Protocols. We also believe it will provide an enjoyable and interesting introduction to the field of protocol design as a whole and so will potentially influence students to continue study in this area at a tertiary level.

We have some constructive feedback on this chapter already so the immediate future work for this content would be to improve the IRC section, and point out the benefits of custom Packet Attack levels for asking "what if" questions. We also hope to look at student answers in 2016, where we can really see any effect this content had on the quantity and quality of answers on the topic, and adjust it to address any misconceptions that come through.

Due to this work, and other work like it [13, 19, 6, 4] students are starting to gain a more accurate perception of Computer Science, have a more positive experience and are choosing it as a career path. Hopefully this leads to more graduates in a few years and not as many roles going unfilled in our increasingly digital world.

Bibliography

[1] Audience retention report - youtube help. https://support.google.com/youtube/answer/1715160?hl=en. (Visited on 10/10/2014).

[2] Tim Bell, Jack Morgan, and Caitlin Duncan. Computer Science Field Guide. URL http://www.cosc.canterbury.ac.nz/csfieldguide/.

[3] Tim Bell, Peter Andreae, and Anthony Robins. A case study of the Introduction of Computer Science in NZ Schools, 2013.

[4] Tim Bell, Caitlin Duncan, Sam Jarman, and Heidi Newton. Presenting Computer Science Concepts to High School Students. *International Olympiad in Informatics*, 8:3–19, 2014. URL http://www.mii.lt/olympiads_in_informatics/files/volume8.pdf#page=5http://www.mclibre.org/descargar/docs/revista-oii/oii-08_201407.pdf#page=5.

[5] Tim Bell, Caitlin Duncan, Sam Jarman, and Heidi Newton. Presenting Computer Science Concepts to High School Students. *Olympiads in Informatics*, 8:3–19, 2014.

[6] Tim Bell, Heidi Newton, Caitlin Duncan, and Sam Jarman. Adoption of Computer Science in NZ schools. In *Proceedings of ITx - New Zealand's Conference of IT*, pages 203–209, 2014. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:No+Title#0.

[7] Paul Brislen. Ict has no mana: Biggs. 2005. URL http://www.computerworld.co.nz/article/499858/_ict_has_no_mana_biggs/.

[8] T Carrell, V Gough-Jones, and Karen Fahy. Science and Digital Technologies in New Zealand secondary schools: Issues of 21st teaching and learning, senior courses and suitable assessments. (August), 2008. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:The+future+of+Computer+Science+and+Digital+Technologies+in+New+Zealand+secondary+schools+:+Issues+of+21st+teaching+and+learning+,+senior+courses+and+suitable+assessments#5.

[9] Tim Carrell, Vilna GoughJones, and Karen Fahy. The future of Computer Science and Digital Technologies in New Zealand secondary schools: Issues of 21st teaching and learning, senior courses and suitable assessments, August 2008. http://dtg.tki.org.nz/content/download/670/3222/file/Digital%20Technologies%20discussion%20paper.pdf.

[10] Tony Clear and Graham Bidois. Fluency in information technology-fitnz: an ict curriculum meta-framework for new zealand high schools. 2005.

[11] Paul Curzon. Cs4Fn and Computational Thinking Unplugged. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*, WiPSE '13, pages 47–50, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2455-7. doi: 10.1145/2532748.2611263. URL http://doi.acm.org/10.1145/2532748.2611263.

[12] Quintin I. Cutts, Margaret I. Brown, Lynsey Kemp, and Calum Matheson. Enthusing and informing potential computer science students and their teachers. *ACM SIGCSE Bulletin*, 39:196, 2007. ISSN 00978418. doi: 10.1145/1269900.1268842.

[13] Caitlin Duncan. Developing an Online Algorithms Tutorial for New Zealand High Schools. Technical Report February 2011, 2013. URL http://cosc.canterbury.ac.nz/research/reports/HonsReps/2013/hons_1301.pdf.

[14] Y Feaster, F Ali, J Zhai, and JO Hallstrom. Serious toys II: teaching networks, protocols, and algorithms. *Proceedings of the 18th ACM conference on Innovation and Technology in Computer Science Education*, page 143, 2013. doi: 10.1145/1734263.1734313. URL http://portal.acm.org/citation.cfm?doid=1734263.1734313http://dl.acm.org/citation.cfm?id=1734313http://dl.acm.org/citation.cfm?id=2462502.

[15] Stefan Freischlad. Exploration module for understanding the functionality of the internet in secondary education. *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research*, 2007. URL http://dl.acm.org/citation.cfm?id=2449347.

[16] Ben Gibson and Tim Bell. Evaluation of games for teaching computer science. *Proceedings of the 8th Workshop in Primary and Secondary Computing Education — WiPSE '13*, pages 51–60, 2013. doi: 10.1145/2532748.2532751. URL http://dl.acm.org/citation.cfm?doid=2532748.2532751.

[17] A Gramm, M Hornung, and H Witten. Email for you (only?): design and implementation of a context-based learning process on internetworking and cryptography. *Proceedings of the 7th Workshop in Primary and Secondary Computing Education*, pages 116–124, 2012. URL https://www.dropbox.com/sh/628r06wzowu1py6/4bA9m7cJ3PGrammHornungWitten2012EmailforYouonly-2184385024/GrammHornungWitten2012EmailforYouonly.pdfhttp://dl.acm.org/citation.cfm?id=2481477.

[18] Gordon Grimsey and Margot Phillipps. Evaluation of Technology Achievement Standards for use in New Zealand Secondary School Computing Education: A critical report, April 2008. http://www.iitp.org.nz/news/uploads/PDFs/200805NCEAReport.pdf.

[19] Sam Jarman and Tim Bell. A Game to Teach Network Communication Reliability Problems and Solutions. In *The 9th Workshop in Primary and Secondary Computing Education — WiPSCE '14*, page To Appear, 2014.

[20] Jane Margolis and Allan Fisher. *Unlocking the clubhouse: Women in computing*. The MIT Press, 2003.

[21] JN Matthews. Hands-on approach to teaching computer networking using packet traces. *Proceedings of the 6th conference on Information Technology Education*, page 361, 2005. doi: 10.1145/1095714.1095777. URL http://portal.acm.org/citation.cfm?doid=1095714.1095777http://dl.acm.org/citation.cfm?id=1095777.

[22] AM Mesaros and Ira Diethelm. Ways of planning lessons on the topic of networks and the internet. *Proceedings of the 7th Workshop in Primary and Secondary Computing Education*, page 70, 2012. doi: 10.1145/2481449.2481465. URL http://dl.acm.org/citation.cfm?doid=2481449.2481465http://dl.acm.org/citation.cfm?id=2481465.

[23] Brad Richards. Bugs as features: teaching network protocols through debugging. *ACM SIGCSE Bulletin*, pages 3–6, 2000. URL http://dl.acm.org/citation.cfm?id=331865.

[24] Kirstin Schwidrowski. A Catalogue of Exercise Classes for Internetworking.

[25] Kirstin Schwidrowski. A catalogue of exercise classes for internetworking. *IFIP International Federation for Information Processing*, 281:157–160, 2010. URL http://link.springer.com/chapter/10.1007/978-0-387-09729-9_23http://dl.ifip.org/index.php/ifip/article/view/13500.

[26] JD Sloan and Charles Schlindwein. TCP/IP laboratory exercises for use with a remotely accessible networking laboratory. *Journal of Computing Sciences in Colleges*, pages 68–78, 2004. URL http://dl.acm.org/citation.cfm?id=948841.

[27] David Thompson, Tim Bell, Peter Andreae, and Anthony Robins. The Role of Teachers in Implementing Curriculum Changes. In *Proceeding of the 44th ACM technical symposium on Computer science education*, SIGCSE '13, pages 245–250, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1868-6. doi: 10.1145/2445196.2445272. URL http://doi.acm.org/10.1145/2445196.2445272.

[28] A Zengin and H Sarjoughian. Devs-suite simulator: a tool teaching network protocols. *WSC '10: Proceedings of the Winter Simulation Conference*, 2010. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5678989.

A Appendix

A.1 UC Tablets of Stone

UC Tablets of Stone is an adaptation of an activity from the Computer Science Inside project[1]. The UC version is available at `http://www.cosc.canterbury.ac.nz/csfieldguide/dev/dev/_static/files/UCTabletsofStone.pdf` and is also attached in this appendix.

[1] `http://csi.dcs.gla.ac.uk/workshop-view.php?workshopID=4`

Tablets of Stone

In this activity students consider how different methods of communication operate successfully. By looking at rules and procedures in pace students are introduced to communications protocols. By working through a role play scenario pupils test their own protocol operating in an unreliable environment similar to that found in packet switching on the Internet, particularly TCP/ IP.

Aims:
After the activity students should:

- Be able to identify different means of communication and the associated networks they frequently make use of
- Be able to identify and explain the kinds of rules/ protocols for communication that are in place in these networks
- Understand the terms “protocol” and “communication” and be able to explain them
- Understand the importance of reliability in a network and where it is and is not achieved in networks they are familiar with
- Successfully analyse a problem specification and draw out the important issues involved
- Be able to consider different methods of achieving reliable communication
- Understand the need for an agreed upon protocol for communication
- Understand the impact on the sender/ receiver and message if the protocol is not complete or correctly adhered to
- Have knowledge to develop their own protocol with the aim of achieving reliable communication
- Be able to identify the protocol information that is needed to enable communication over an unreliable network
- Be able to follow the rules of a protocol they have created
- Discuss potential flaws in the protocols they have developed and suggest measures to overcome them
- Apply their protocol to other instances of communication they use, such as a mobile phone or the Internet.

Preparation (30 minutes)

1. First gather the cards. You’ll need to print out the action cards (below) and cut them up. These form the basis of the game.
2. Next, decide on some messages for student to send. It’s important that they’re *not* English sentences or anything that can be put back together by their structure. Something like “1LHC255HD(RLLS” would be a suitable message, or a phone number.
3. Print out copies of the "tablets" . Each tablet has places for six characters or numbers, so you cannot fit the whole message on one tablet.

Note: The action cards are three types; delay, don’t deliver, deliver. Adjusting the ratio between these will represent the quality of your messengers. More "deliver" cards means a more reliable messenger. More "delay" and "don’t delivers means a less reliable network. These cards are analogous to a computer network/communication channel.

Playing the game

1. Split your class into pairs. It is crucial for the pairs to sit apart from one another where they cannot see or communicate with each other. Two rooms are ideal but sitting students on opposing sides of a classroom should suffice.
2. Give one of each pair a message to deliver to their partner.
3. Shuffle the Action Cards and choose a messenger. You could be the messenger or if you use a student if you have an odd number. You might need more than one messenger if you have a large class.
4. A student is now to write on their tablet and give it to the messenger. The tablet should at least say the name of the other person on it.
5. The messenger now picks the top action card, turns it over, reads it and uses it to decide what to do with the tablet.
6. Repeat steps 4 and 5 with each tablet.

After 5 or so minutes of chaos and frustration, your students should realise that names alone are not good enough for a protocol. Stop the class and discuss this... what is the first issue they're having? Is it order? Perhaps it would be be best to use one of those 6 slots to put a tablet number in? This means there is less room for the actual data - what does this mean in terms of the number of tablets we have to use now?

After some more time, they might notice other problems, and these should also be discussed. Possible problems could be missing tablet, not knowing if the tablet was delivered, not knowing whether to resend a tablet. Solutions you could suggest would be a sending back acknowledgements and waiting to hear back for these before re-sending another - this means that the receiving student(s) also need blank tablets to send messages, and they will have to agree on what their 6-character responses mean before they play the game again.

You'll need at least two students for this game, but we recommend having as many as possible. If you have a large class, consider a few messengers. Once again, discuss this with your class... what happens if you have many messengers? What happens if you had one?

Deliver this tablet now	Deliver this message after the next one
Deliver this tablet now	Deliver this message after the next one
Deliver this tablet now	Deliver this message after the next one
Deliver this tablet now	Don’t deliver this message
Deliver this tablet now	Don’t deliver this message

To: From:	To: From:
To: From:	To: From:
To: From:	To: From:
To: From:	To: From:

Tablets of Stone

In an ancient city there are a number of very important Governors. These Governors decide how the city is run and make very important decisions. They each live in different houses all over the city.

The Governors often want to communicate, they need to send and receive messages all over the city. Governors are identified by their house number and they all have access to a group of messengers whose job it is to deliver the messages.

The only way to send messages is by writing them on large rectangular stone tablets which the messengers carry to their destination. The stone tablets are of a fixed size and can only fit 6 pieces of information on them. One piece of information can be one letter or one number. Messages are often split over a number of tablets, and as these tablets are very heavy they can only be carried one at a time.

The messengers cannot be trusted to always deliver the message correctly as they are forgetful and lazy. They often stop for long breaks during working hours and even try to escape from the city.

The Governors want to find a way of making their communication reliable, they want to develop a set of rules that they will all follow. By doing this they can tell whether or not their message has been delivered and if the message was correct. The Governors have already decided that the destination should be written on the tablet.

In your groups your task is develop the rules that the Governors will use to communicate...

A.2 The Network Protocols Chapter

The *teacher version* of the Network Communication Protocols chapter of the CSFG is available, and is best viewed, at
`http://bit.ly/csfgteacher`. However, the URL for the teachers version changes every major release. The student/public version is available at `www.csfieldguide.org.nz` and includes information to acess the teachers version. A PDF version is provided in this appendix, captured 14th October. Note that constant updates are part of the project, so the chapter is likely to change and improve. Also note this is the teacher version (teacher notes are included, a chevron precedes these sections).

15. NETWORK COMMUNICATION PROTOCOLS

For teachers

This chapter is under evaluation. It covers one of six topics that students must select two from for the 3.44 Achievement Standard; the other 5 topics have chapters available, so there is still a wide range of choice available. Any feedback would be greatly appreciated. If you also teach your class, please complete the teacher survey here (http://bit.ly/NCPSurveyTeachers) and and your students can do this one http://bit.ly/NCPSurveyStudents (http://bit.ly/NCPSurveyStudents)

For teachers

See the URL above you? Notice how it says “http:// (http://)” the p stands for protocol. They’re all around us on the web and this chapter will teach you more.

This chapter supports the “Network communication protocols” option of the NZ achievement standard 3.44.

Currently all material in this chapter is relevant to the standard, although students can choose one or two examples to focus on to meet the requirements of the standard.

Be careful not to confuse this topic with those from the infrastructure standards such as the 2.50 or 2.51 achievement standards. 2.50 ensures a student knows the layers in the TCP/IP networking model and the role of this model in LAN architecture. We approach this from a different perspective, not touching on the OSI model, but mainly acknowledging the existence of a layered model. Similar approaches at level three in 2.50 with WAN architecture. This chapter focuses on the problems and the techniques used to solve them from a computer science perspective. While there is overlap, high achieving submissions will also take this approach.

For teachers

An alternative video could be this one from Family Guy: https://www.youtube.com/watch?v=KJCfUm21BsI (https://www.youtube.com/watch?v=KJCfUm21BsI). However, it might not be appropriate for your school, as it does contain some questionable language. A vimeo version is at http://vimeo.com/106027170 (http://vimeo.com/106027170). **Network communication protocols** focus on the techniques applied in computer networks to ensure reliable communication of data between two parts of a network in the face of different kinds of threats and failures. Reliable refers to messages being on time, in order and not damaged. The project in this chapter would typically be done by giving examples of the sequence of events that occur in these situations, discussing how the protocols and their coding schemes overcome the problems, and evaluating how successful they are at addressing them. This topic is distinct from the coverage of networking in the infrastructure standards because it focuses on the issues that the protocols address (i.e. the design of the protocol), rather than how to configure a system that uses a given protocol.

15.1. WHAT'S THE BIG PICTURE?

Think about the last time someone sent you mail via the post. They probably wrote some content on some paper, put it in an envelope, wrote an address and put it in a postbox. From there, the letter probably went into a sorting center, got sorted, and was put in a bag. The bag then went into a vehicle like a truck, plane or boat. The vehicle either travelled through water, the air, or on the road. The postal system is a complicated one, designed to let individuals communicate easily, yet being efficient enough to group many letters into one postal delivery. The same ideas apply to how messages move around the internet. Whether it be a 'like' on Facebook, a video stream or an email - the internet and its various protocols looks after it for you so it is delivered on time and intact to the other person.

Below we introduce some concepts, algorithms, techniques, applications and problems that relate to network protocols; it isn't a complete list of all the ideas in the area, but should be enough to give you a good idea of what this area of computer science is about.

For teachers

Key concepts that are likely to be encountered are: formal communication, structured requests/responses, addressing, packet loss, quality of service, network performance.

Algorithms: (techniques are more relevant to this area than algorithms)

Techniques: packet switching, handshaking, acknowledgement, checksums, redundancy, packet ordering and use of timeouts.

Applications: Protocols that you will investigate are HTTP and IRC, and UDP and TCP. You could also look at DNS, FTP, DHCP, Telnet or SSH. The applications of network communication protocols are easier when you use those from the Application level (rather than the transport or internet), however UDP and TCP are more complex and more interesting.

For teachers

Running the Tablets of Stone game (below) can take some time and coordination to do properly. The game is meant to cause frustration, and is usually not finished in an a typical hour long class. The game is best when played for a short period (5 minutes or so) and then stopped and discussed. Discuss the issues the students are facing, packet loss, packet delay, etc. Then begin to discuss possible solutions. These could be packet numbering, positive or negative acknowledgments, timeouts etc. However, this may take more time, and also lets you fit less information on each tablet. Students should realise this tradeoff and understand the key problems of network communication protocols as efficiency and reliability when communicating. Tablets of Stone can be found at University of Glasgow Computer Science Department website (http://csi.dcs.gla.ac.uk/workshop-view.php?workshopID=4). We've made a guide too, which you can download freely here (_static/files/UCTabletsofStone.pdf). Other activities you should consider are "Locked-In Syndrome, described here (http://dl.acm.org/citation.cfm?doid=2532748.2611263) or Code.Org's *The Internet* activity, described here (https://learn.code.org/s/1/level/102).

15.2. GETTING STARTED

Take part in a game of Tablets of Stone in your classroom. Your teacher will show you how it is played. Try to think about a few things while you're playing the game. What happens if one of my messages is delayed? What happens if one of my messages gets lost completely? Will the other governor be able to put them back together?

15.3. WHAT IS A PROTOCOL?

'Protocol' is a fancy word for simply saying "an agreed way to do something". You might have heard it in a cheesy cop show – "argh Jim, that's against protocol!!!" – or heard it used in a procedural sense, such as how to file a tax return or sit a driving test. We all use protocols, every day. Think of when you're in class. The *protocol* for asking a question may be as follows: raise your hand, wait for a nod from the teacher then begin asking your question.

Simple tasks require simple protocols like the one above; however more complicated processes may require more complicated protocols. Pilots and aviation crew have their own language (almost) for their tasks. A subset of normal language used to convey information such as altitude, heading, people on board, status and more.

Activities on the internet vary a lot too (email, skype, video streaming, music, gaming, browsing, chatting), and so do the protocols used to achieve these. These collections of protocols form the topic of Networking Communication Protocols and this chapter will introduce you to some of them, what problems they solve, and what you can do to experience these protocols first hand. Let's start by talking about the one you're using if you're viewing this page on the web.

15.4. APPLICATION LEVEL PROTOCOLS - HTTP, IRC

The URL for the home site of this book is http://csfieldguide.org (http://csfieldguide.org). Ask a few friends what the "http" stands for - they have probably seen it thousands of times...do they know what it is? This section covers high level protocols such as HTTP and IRC, what they can do and how you can use them (hint: you're already using HTTP right now).

For teachers

For the activity in this section it's ideal if your school computers have a modern browser with the developer extensions enabled. Chrome (https://www.google.com/chrome/browser/) is free to download. Follow the instructions here (http://debugbrowser.com/) for more information on how to do this. The developer browser can't really do any harm, and you can encourage further tinkering. However, knowledge of HTML, JavaScript or any other web design won't be helpful in a report on protocols. If your school's IT department can't allow you access to these features, simply encourage the students to try this at home. It's a perfectly safe activity. Note: Details of page loads only show up if the inspector is open, you may need to refresh the current page to see this.

15.4.1. HYPERTEXT TRANSFER PROTOCOL (HTTP)

The HyperText Transfer Protocol (HTTP) is the most common protocol in use on the internet. The protocol's job is to transfer *HyperText* (such as HTML) from a server to your computer. It's doing that right now. You just loaded the Field Guide from the servers where it is hosted. Hit refresh and you'll see it in action.

HTTP functions as a simple conversation between client and server. Think of when you're at a shop:

You: "Can I have a can of soda please?"
Shop Keeper: "Sure, here's a can of soda"

HTTP uses a request/response pattern for solving the problem of reliable communication between client and server. The "ask for" is known as a *request* and the reply is known as a *response*. Both requests and responses can also have other data or *resources* sent along with it.

Curiosity

A *resource* is any item of data on a server. For example, a blog post, a customer, an item of stock or a news story. As a business or a website, you would create, read, update and delete these as part of your daily business. HTTP is well suited to that. For example, if you're a news site, every day your authors would add stories, you could update them, delete them if they're old or become out of date, all sorts. These sorts of methods are required to manage content on a server, and HTTP is the way to do this.

This is happening all the time when you're browsing the web; every web page you look at is delivered using the HyperText Transfer Protocol. Going back to the shop analogy, consider the same example, this time with more resources shown in asterisk (*) characters.

You: "Can I have a can of soda please?" *You hand the shop keeper $2*
Shop Keeper: "Sure, here's a can of soda" *Also hands you a receipt and your change*

There are nine *types* of requests that HTTP supports, and these are outlined below.

A GET request returns some text that describes the thing you're asking for. Like above, you ask for a can of soda, you get a can of soda.

A HEAD request returns what you'd get if you did a GET request. It's like this:

You: "Can I have a can of soda please?"
Shop Keeper: "Sure, here's the can of soda you'd get" *Holds up a can of soda*

What's neat about HTTP is that it allows you to also modify the contents of the server. Say you're now also a representative for the soda company, and you'd like to re-stock some shelves.

A POST request allows you to send information in the other direction. This request allows you to replace a resource on the server with one you supply. These use what is called a Uniform Resource Identifier or URI. A URI is a unique code or number for a resource. Confused? Let's go back to the shop:

Sales Rep: "I'd like to replace this dented can of soda with barcode number 123-111-221 with this one, that isn't dented"
Shop Keeper: "Sure, that has now been replaced"

A PUT request adds a new resource to a server, however, if the resource already exists with that URI, it is modified with the new one.

Sales Rep: "Here, have 10 more cans of lemonade for this shelf"
Shop Keeper: "Thanks, I've now put them on the shelf"

A DELETE request does what you'd think, it deletes a resource.

Sales Rep: "We are no longer selling 'Lemonade with Extra Vegetables', no one likes it! Please

remove them!”
Shop Keeper: “Okay, they are gone”.

Some other request types (*HTTP methods*) exist too, but they are less used; these are TRACE, OPTIONS, CONNECT and PATCH. You can find out more about these (http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) on your own if you’re interested.

In HTTP, the first line of the response is called the *status* line and has a numeric status code such as **404** and a text-based *reason phrase* such as “Not Found”. The most common is 200 and this means successful or “OK”. HTTP status codes are primarily divided into five groups for better explanation of requests and responses between client and server and are named by purpose and a number: Informational 1XX, Successful 2XX, Redirection 3XX, Client Error 4XX and Server Error 5XX. There are many status codes (http://en.wikipedia.org/wiki/List_of_HTTP_status_codes) for representing different cases for error or success. There’s even a nice 418: Teapot error on Google: http://www.google.com/teapot (http://www.google.com/teapot)

For teachers

Chrome (https://www.google.com/chrome/browser/) is free to download. Follow the instructions here (http://debugbrowser.com/) for the browser you have installed.

So what’s actually happening? Well, let’s find out. If you’re in a Chrome or Safari browser, press Ctrl + Shift + I in windows or Command + Option + I on a mac to bring up the web inspector. Select the Network tab. Refresh the page. What you’re seeing now is a list of of HTTP requests your browser is making to the server to load the page you’re currently viewing. Near the top you’ll see a request to NetworkCommunicationProtocols.html. Click that and you’ll see details of the Headers, Preview, Response, Cookies and Timing. Ignore those last two for now.

Let’s look at the first few lines of the headers:

```
Remote Address:132.181.17.3:80
Request URL:http://csfieldguide.org.nz/NetworkCommunicationProtocols.html
Request Method:GET
Status Code:200 OK
```

The *Remote Address* is the address of the server of the page is hosted on. The *Request URL* is the original URL that you requested. The request method should be familiar from above. It is a GET type request, saying “can I have the web page please?” and the response is the HTML. Don’t believe me? Click the *Response* tab. Finally, the *Status Code* is a code that the page can respond with.

Let’s look at the *Request Headers* now, click ‘view source’ to see the original request.

```
GET /NetworkCommunicationProtocols.html HTTP/1.1
Host: www.cosc.canterbury.ac.nz
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=
0.8
User-Agent: Chrome/34.0.1847.116
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
```

As you can see, a request message consists of the following:

- A request line in the form of *method URI protocol/version*
- Request Headers (Accept, User-Agent, Accept-Language etc)
- An empty line
- An optional message body.

Let's look at the *Response Headers*:

```
HTTP/1.1 200 OK
Date: Sun, 11 May 2014 03:52:56 GMT
Server: Apache/2.2.15 (Red Hat)
Accept-Ranges: bytes
Content-Length: 3947
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
Vary: Accept-Encoding, User-Agent
Content-Encoding: gzip
```

As you can see, a request message consists of the following:

- Status Line, 200 OK means everything went well.
- Response Headers (Content-Length, Content-Type etc)
- An empty line
- An optional message body.

Go ahead and try this same process on a few other pages too. For example, try these sites:

- A very busy website in terms of content, such as *Facebook.com*
- A chapter that doesn't exist in the Field Guide (http://csfieldguide.org.nz/Nope.html)
- Your favourite website

Curiosity

Tim Berners-Lee was credited for creating HTTP in 1989. You can read more about him here (http://en.wikipedia.org/wiki/Tim_Berners-Lee).

15.4.2. INTERNET RELAY CHAT (IRC)

For teachers

For this section we suggest you use the freenode web client. (http://webchat.freenode.net/) This allows you to set up your own channel and then your students can join it too. Now, with channels on this service, anyone can join them, so if your channel name is too obvious, you might get members of the public joining. It's best to break convention with channel names and use ###irc-myschool-thedate or something similar. Students can also download and install IRC clients, but this is complicated to configure, so it's best to use the web version for now. Just tell them what channel to join.

Internet Relay Chat (IRC) is a system that lets you transfer messages in the form of text. It's essentially a chat protocol. The system uses a client-server model. Clients are chat programs installed on a user's computer that connect to a central server. The clients communicate the message to the central server which in turn relays that to other clients. The protocol was originally designed for group communication in a discussion forum, called *channels*. IRC also supports one-to-one communication via *private messages*. It is also capable of file and data transfer too.

The neat thing about IRC is that users can use commands to interact with the server, client or other users. For example /DIE will tell the server to shutdown (although it will only work if you are the administrator!) /ADMIN will tell you who the administrator is.

Whilst IRC may be new to you, the concept of a group conversation online or a *chat room* may not be. There really isn't any difference. Groups exist in the forms of *channels*. A server hosts many channels, and you can choose which one to join.

Channels usually form around a particular topic, such as Python, Music, TV show fans, Gaming or Hacking. Convention dictates that channel names start with one or two # symbols, such as #python or ##TheBigBangTheory. *Conventions* are different to protocols in that they aren't actually enforced by the protocol, but people choose to use it that way.

To get started with IRC, first you should get a client. A client is a program that let's you connect Ask your teacher about which one to use. For this chapter, we'll use the freenode web client. (http://webchat.freenode.net/). Check with your teacher about which channel to join, as they may have set one up for you.

Try a few things while you're in there. Look at this list of commands (http://en.wikipedia.org/wiki/List_of_Internet_Relay_Chat_commands) and try to use some of them. What response do you get? Does this make sense?

Try a one on one conversation with a friend. If they use commands, do you see them? How about the other way around?

15.5. PROJECTS - HTTP AND IRC

For teachers

As further guidance as to what students should do, we have made this overall outline of how the project fits the standard. Note that for 3.44 this covers one of the two areas of computer science students have to cover in their report. They must pick one other area of computer science to do a project on as well.

Selected Area: Network Communication Protocols

Key Problems:

- The bigger problems are: Reliable, efficient communication suitable to purpose. Abstracted level of communication
- Application layer: Transfer of text, video, audio, files. Effective error communication.

Key algorithm/Techniques:

- Bigger solutions: A layered network model approach, each level having its role and abstracting it from the level above.
- Application layer: requests, responses, resources, error codes, commands.

Examples of practical applications: Looking at web traffic with the network inspector/developer tools or using a private IRC channel and using some commands.

Personalised student examples: Each website a student visits should be unique, they should report on unique requests and responses they see. They could talk about different error codes and why they exist in groups (100, 200, 300, 400 and 500). With IRC, a *sensible* chat session could be recorded and experiences with various commands.

For achieved, it could be expected that a student...

"describing key problems that are addressed in selected areas of computer science"

- Describing the problems at the overall level, or on the application layer.

"describing examples of practical applications of selected areas to demonstrate the use of key algorithms and/or techniques from these areas"

- Carrying out an activities above
- Describing how the activity went
- Reporting on the results
- Describing at least 2 or 3 samples of the activity
- Showing data/results from those runs
- Describe one technique that is used in this activity

For merit, it could be expected that a student…

"explaining how key algorithms or techniques are applied in selected areas"

- Explaining what factors have to be considered when carrying out the the activity, to ensure a valid result.
- Explaining what HTTP or IRC achieves and how it achieves that.

"explaining examples of practical applications of selected areas to demonstrate the use of key algorithms and/or techniques from these areas."

- Explaining how the the activity could be used to evaluate a new protocol
- Giving examples of the protocol trying to address these issues

For excellence, it could be expected that a student…

"discussing examples of practical applications of selected areas to demonstrate the use of key algorithms and/or techniques from these areas"

- Discussing other possible applications of techniques used in HTTP or IRC. This may be difficult and not suitable for excellence level.

"evaluating the effectiveness of algorithms, techniques, or applications from selected areas."

- Discussing the various protocols, how they are used and why.
- Discussing the differences of HTTP and HTTPS. Discussing IRC vs another chat protocol (telnet, linux's wall).

HTTP is the most common protocol yet. We use it every day and you're using it right now if you're viewing this on the web. Open up the web inspector (you might have to do this at home if your school doesn't have it available) and have a look at the traffic. You might need to refresh, depending on your browser. We recommend Chrome, which is free to download at https://www.google.com/chrome/browser/ (https://www.google.com/chrome/browser/).

Bring up the Developer Tools. Find the *Network* tab. Reload the page. You should now see a slew of request form. Go through and click each one. Have a look at the details you can see under headers.

- What is the remote address?
- What type of method is it?
- What does the status code mean?
- What is the path?
- What is the response?
- What is *actually happening* here? What part of the page is loading?

For teachers

Here's a quick video from Numb3rs -"how hackers talk when they don't want to be overheard" (https://www.youtube.com/watch?v=O2rGTXHvPCQ)

IRC is a very primitive chat program which is fun to use with your friends.

Go be a hacker, grab a friend and visit freenode and create a channel for you. (http://webchat.freenode.net/) Now, look at the list of commands you can use (http://en.wikipedia.org/wiki/List_of_Internet_Relay_Chat_commands) and try to use some of them. What response do you get? Does this make sense?

Try a one on one conversation with a friend. If they use commands, do you see them? How about the other way around?

For teachers

TCP and UDP are a lot more complex than the protocols above. However are required to get the depth needed for Excellence and Merit. TCP is a protocol that addresses packet loss, duplication, corruption and delay, UDP does not. Packet reliability is fixed for a chosen set of solutions in TCP, in UDP it is not. Because these solutions takes time, TCP is typically not suitable for real-time communication such as Skype, music or video or online video gaming, but UDP is not suitable for email, text, downloads etc - times where all data has to be present and accounted for.

15.6. TRANSPORT LAYER PROTOCOLS - TCP AND UDP

So far we have talked about HTTP and IRC. These protocols are at a level that make sure you do not need to worry about how your data is being transported. Now we'll cover how your data is transferred reliably and efficiently, regardless of what the data is. Below this level is an unreliable medium for transfer (such as wifi or cable, which are subject to interference errors) which causes a concern for data transportation. These protocols take different approaches to ensure data is delivered in an effective and/or efficient manner.

15.6.1. TCP

TCP (The Transmission Control Protocol) is one of the most important protocol on the internet. It breaks large messages up into *packets*. What is a packet? A packet is a segment of data that when combined with other packets, make up a total message (something like a HTTP request, an email, an IRC message or a file like a picture or song being downloaded). For the rest of the section, we'll look at how these are used to load an image from a website.

So computer A looks the file and takes it, breaks it into packets. It then sends the packets over the internet and computer B reassembles them and gives them back to you as the image, which is demonstrated in this video. (https://www.youtube.com/watch?v=WwyJGzZmBe8)

By now you're probably wondering why we bother splitting up packets… wouldn't it be easier to send the file as a whole? Well, it solves congestion. Imagine you're in a bus, in rush hour and you have to be home by 5. The road is jammed and there's no way you and your friends are getting home on time. So you decide to get out of the bus and go your own separate ways. Web pages are like this too. They are too big to travel together so they are split up and sent in tiny pieces and then reassembled at the other end.

So why don't the packets all just go from computer A to computer B just fine? Ha! That'd be nice. Unfortunately it's not that simple. Through various means, there are some problems that can affect packets. These problems are:

- Packet loss
- Packet delay (packets arrive out of order)
- Packet corruption (the packet gets changed on the way)

So, if we didn't try fix these, the image wouldn't load, bits would be missing, corrupted or computer B might not even recognise what it is!

So, TCP is a protocol that solves these issues. To introduce you to TCP, please play the game below. In the game, *you* are the problems (loss, delay, corruption) and as you move through the levels, pay attention to how the computer tries to combat them. Good luck trying to stop the messages getting through correctly!

For teachers

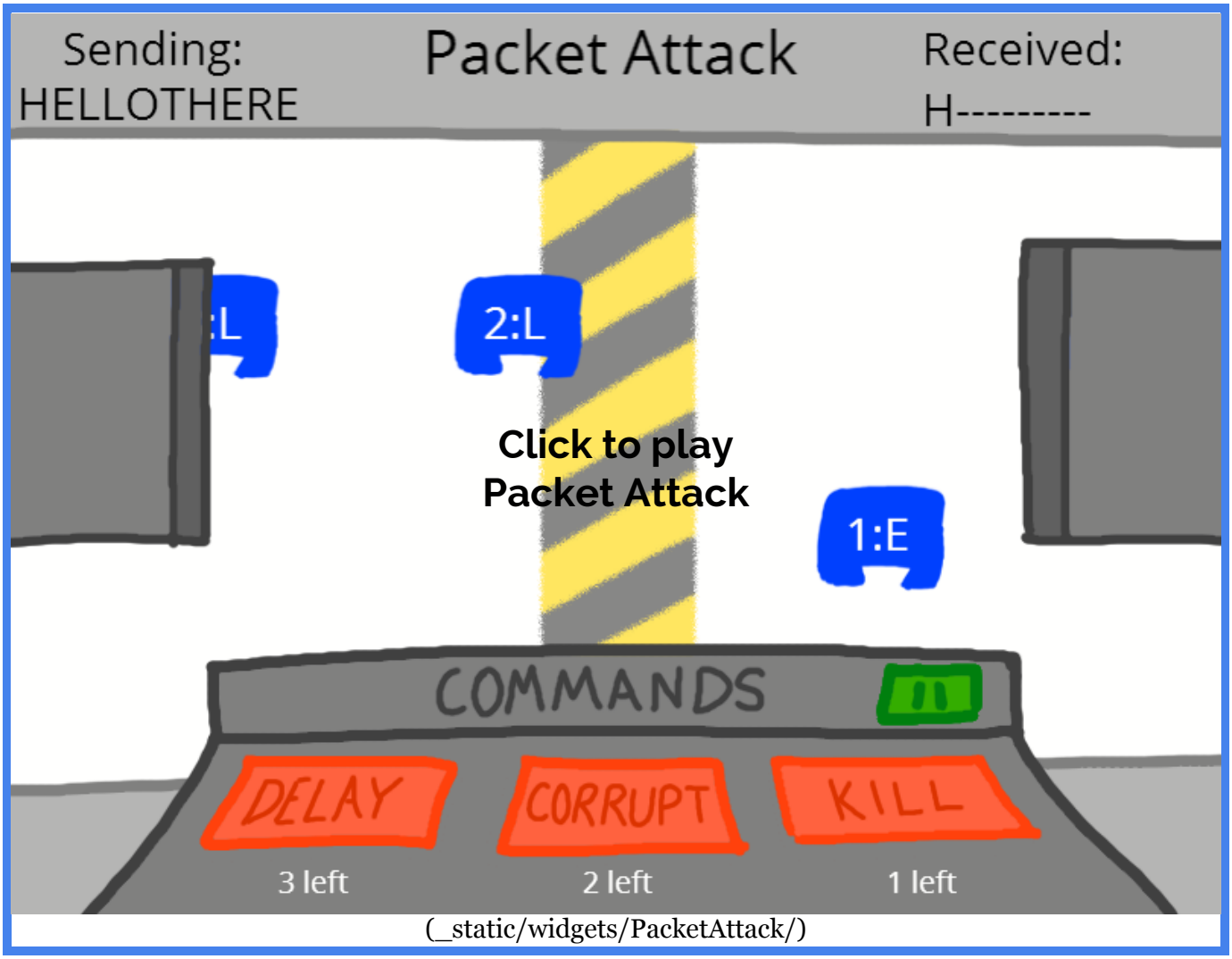
You can also create your own 'levels' for your students in Packet Attack. We've also put a level builder in the projects section below so that you can make levels for your students, experiment with different reliabilities or combination of defenses.

Adjust the trues, falses and numbers to set different abilities. Raising the numbers will effectively equate to a less reliable communication channel. Adding in more abilities (by setting shields etc to true) will make for a harder to level to beat.

Level Overview

- *Level 1*: No defences. One packet. Students can beat by using corrupt or kill.
- *Level 2*: Multiple Packets. 10 packets. Students can't stop all packets, but can corrupt and kill and delay for points.
- *Level 3*: Shields enabled. 10 packets. Students can beat by using kill, but corrupt won't work.
- *Level 4*: Numbering enabled. 10 packets. Students can beat by using kill, corrupt, but delay won't work.
- *Level 5*: Numbers and shields. 10 packets.

- *Level 6*: Numbers and acknowledgments. Packets will be sent back and resent.
- *Level 7*: Numbers, shields, timeouts and acknowledgments. Packets will be sent back and resent. This level is not beatable.



(_static/widgets/PacketAttack/)

For teachers

Packet Attack is a direct analogy for TCP and it is intended to be an interactive simulation of it. The Packet Creatures are TCP segments, which move between two computers. The yellow/gray zone is the unreliable channel, susceptible to unreliability. That unreliability is the user playing the game. Remember from the key problems of this topic on the transport level, we have delays, corruption and lost packets, these are the attacks; *delay*, *corrupt*, *kill*. Solutions come in the form of TCP mechanisms, they are slowly added level by level. Like in TCP, the game supports packet ordering, checksums (shields), Acks and Nacks (return creatures) and timeouts .

Let's talk about what you saw in that game. What did the levels do to solve the issues of packet loss, delay (reordering) and corruption? TCP has several mechanisms for dealing with packet troubles.

Curiosity

Why do packets experience delays, loss and corruption? This is because as packets are sent over a network, they go through various *nodes*. These nodes are effectively different routers or computers. One route might experience more interference than another (causing packet loss), one might be faster or shorter than another (causing order to be lost). Corruption can happen at any time through electronic interference.

Firstly, TCP starts by doing what is known as a handshake. This basically means the two computers say to each other: “Hey, we’re going to use TCP for this image. Reconstruct it as you would”.

Next is **Ordering**. Since a computer can’t look at data and order it like we can (like when we do a jigsaw puzzle or play Scrabble™) they need a way to “stitch” the packets back together. As we saw in *Packet Attack*, if you delayed a message that didn’t have ordering, the message may look like “HELOLWOLRD”. So, TCP puts a number on each packet (called a sequence number) which signifies its order. With this, it can put them back together again. It’s a bit like when you print out a few pages from a printer and you see “*Page 2 of 11*” on the bottom. Now, if packets do become out of order, TCP will wait for all of the packets to arrive and then put the message together.

Another concept is **checksums**. This concept of storing information about the data may be familiar from the error control coding chapter (http://www.cosc.canterbury.ac.nz/csfieldguide/ErrorControlCoding.html). Basically, a checksum can detect errors and sometimes with coding schemes, can correct them. In the case of a correctable packet, it is corrected. If not, the packet is useless and needs to be resent. In the game, shields represent checksums. Corrupt a checksum once, and it can recover from the error using error correction. Corrupt it again and it can’t.

So how do packets get re-sent? TCP has a concept of *acknowledgement* and *negative acknowledgement* messages (ACK and NACK for short). You would have seen these in the higher levels of the game as the green (ACK) and red (NACK) creatures going back. Acks are sent to let the sender know when a packet arrives and it is usable. Nacks are sent back when a packet arrives and is damaged and needs resending. ACKs and and NACKs are useful because they provide a channel *in the opposite direction* for communication. If computer A receives a NACK, they can resend the message. If it receives an Ack, the computer can stop worrying about a resend.

But does a computer send it again if it doesn’t hear back? Yes. It’s called a timeout and it’s the final line of defense in TCP. If a computer doesn’t get an ACK or a NACK back, after a certain time it will just resend the packet. It’s a bit like when you’re tuning out in class, and the teacher keeps repeating your name until you answer. Maybe that’s been you… woops. Sometimes, an ACK might get lost, so the packet is resent after a timeout, but that’s OK, as TCP can recognise duplicates and ignore them.

So that's TCP. A protocol that puts accurate data transmission before efficiency and speed in a network. It uses timeouts, checksums, acks and nacks and many packets to deliver a message reliably. However, what if we don't need all the packets? Can we get the overall picture faster? Read on...

15.6.2. UDP

UDP (User Datagram Protocol) is a protocol for sending packets that does not guarantee delivery. UDP doesn't guarantee against lost packets, duplicate packets or out of order packets. It just gets the bulk of the data there when it can. Checksums are used for data integrity though, so they have some protection. It's still a protocol because it has a formal packet structure. The packets still include destination and origin as well as the size of the packet.

So do we even use such an unreliable protocol? Yes, but not for anything too important. Files, messages, emails, web pages and other text based items use TCP, but things like streaming music, video, VOIP and so on use UDP. Maybe you've had a call on Skype that has been poor quality. Maybe the video flickers or the sound drops for a split second. That's packets being lost. However, you of course get the overall picture and the conversation you're having is successful.

15.7. PROJECTS - TCP AND UDP

For teachers

As further guidance as to what students should do, we have made this overall outline of how the project fits the standard. Note that for 3.44 this covers one of the two areas of computer science students have to cover in their report. They must pick one other area of computer science to do a project on as well.

Selected Area: Network Communication Protocols

Key Problems:

- The bigger problems are: Reliable, efficient communication suitable to purpose. Abstracted level of communication
- Application layer: Transfer of text, video, audio, files. Effective error communication.
- Transport layer: Packet loss, packet corruption, packet delay

Key algorithm/Techniques:

- Bigger solutions: A layered network model approach, each level having its role and abstracting it from the level above.
- Application layer: requests, responses, resources, error codes.
- Transport layer: packet switching, handshaking, timeouts, acknowledgements, packet ordering

Examples of practical applications:

Practical applications of TCP are non-realtime applications such as email, web browsing, file transfer and downloading. UDP is more for real time applications such as music and video streaming, online gaming and VoIP and VoLTE. You can't really see these in practice since they're are abstracted away, but you can dig into it for a project with some tools. Wireshark, TCP Dump or Windump (http://technet.microsoft.com/en-us/magazine/2005.05.howitworkstcpip.aspx) Note: WinDump and TCPDump spits out your *own* TCP traffic, so there are no privacy issues there. Traffic can be recreated by adding together TCP packets, so be careful about posting packets online - they may contain sensitive data such as passwords.

Personalised student examples:

Each student experience playing Tablets of Stone or playing Packet Attack should be unique and can be used to illustrate TCP and UDP. Also, using TCPDump or Windump would be a great example, but is a little more difficult. Use the guides we've linked to. It's almost certain students will have to perform these tasks at home.

For achieved, it could be expected that a student…

"describing key problems that are addressed in selected areas of computer science"

- Describing the problems at the overall level, or at application or transport layer

"describing examples of practical applications of selected areas to demonstrate the use of key algorithms and/or techniques from these areas"

- Carrying out an activity above
- Describing how the activity went
- Reporting on the results
- Describing at least 2 or 3 runs of the activity
- Showing any data/results/screenshots/photographs from those runs
- Describe one technique that is used in this activity

Questions:

- What problems did you encounter when transmitting data across networks? Talk about problems that you encountered at the start of Tablets of Stone (photos are great!)
- What solutions did you see in Packet Attack, what attacks did they stop?

For merit, it could be expected that a student…

"explaining how key algorithms or techniques are applied in selected areas"

- Explaining what factors have to be considered when carrying out the the activity, to ensure a valid result.
- Explaining the details of at least two techniques used by lower level protocols to address

reliability or efficiency.
- For example, imagine you had to post a 500 page book, using only postcards. What would you use? How would you correct for lost postcards?

Questions:

- Explain the use of TCP and UDP in networks today, with example situations. What systems use TCP? Which use UDP?
- Explain at least two techniques used by TCP and/or UDP to address the problems above. Show some examples from packet attack (and tablets of stone) that illustrate the concepts.

"explaining examples of practical applications of selected areas to demonstrate the use of key algorithms and/or techniques from these areas."

- Giving examples of the protocol trying to address these issues
- Explain what headers, footers etc do on a packet, and what is in them
- Explain what TCP is used for and what UDP is use for and why (real time vs accuracy)

For excellence, it could be expected that a student...

"discussing examples of practical applications of selected areas to demonstrate the use of key algorithms and/or techniques from these areas"

- Discussing other possible applications of techniques used in Packet Attack and Tablets of Stone that display TCP and UDP principles.

"evaluating the effectiveness of algorithms, techniques, or applications from selected areas."

- Discussing the various protocols, how they are used and why.
- Explaining how the the activity could be used to evaluate a new protocol
- Discussing the differences of TCP vs UDP and why they exist and/or their applications. (Effectiveness could be reliability or speed - usually a trade off. Packet size? Content vs Correctness)
- In terms of evaluation, students could use a stopwatch to time several runs of Packet Attack or Tablets of Stone and graph the trend of reliability and delivery time with all of TCP in place. For example, they could set up a custom level in packet attack with 5 kills, 10 kills, 20 kills, 50 kills and see if there is a trend. Report on it, discuss why they are seeing it.

Questions:

- For systems that require TCP, what might happen to them if TCP did not exist and they had to use UDP?
- Discuss the differences between TCP and UDP, why each exists and why you would choose a particular protocol for several scenarios.

Before writing about Network Communication Protocols, think about the following questions:

Tablets of Stone:

- How did your messaging go when you first started?
- Did you need numbers on your tablets?
- Was it a pain to use up tablet space on numbers and other information?
- Did you ever get to reliable communication?

Packet Attack:

- What happens if you add too many kills, corrupts and delays? Is there a relationship between this and time taken to transmit the message? Try graphing it
- What happens if you turn off all the defenses?
- What happens if you have no kills, corrupts and delays?
- What happens if you only have delays?
- What happens if you kill a packet creature when it tries to get sent the second time?
- What other situations can you get the protocol in?

PACKET ATTACK LEVEL CREATOR

Check any defenses you want, enter some values for the attacks and click Create Level

- [ ] Shields
- [ ] Numbers
- [ ] Timeouts
- [ ] Return Packet Creatures

Number of Delays:

Number of Delays

Number of Corrupts:

Number of Corrupts

Number of Kills:

Number of Kills

CREATE LEVEL

For teachers

Note that the questions cover the report up to merit level, but if students are aiming for Excellence level you will need to make sure they have covered the criteria in the 3.44 standard (there's more information about this in the teacher notes above).

For a project, using the knowledge you have gained on TCP and UDP, create some custom levels in Packet Attack using the controls just above to create some unique situations that illustrate different aspects of Network Protocols. The following questions will help you to reflect on the issues that you could talk about:

1. What problems did you encounter when transmitting data across networks? Talk about problems that you encountered at the start of Tablets of Stone (photos of examples from various stages of the activity are a great way to illustrate it!)
2. Explain the use of TCP and UDP in networks today, with example situations. Which systems use TCP? Which use UDP?
3. Explain at least two techniques used by TCP and UDP to address the problems above. Show some examples from Packet Attack (and/or Tablets of Stone) that illustrate the concepts.
4. For systems that require TCP, what might happen to them if TCP did not exist and they had to use UDP?
5. Discuss the differences between TCP and UDP, why each exists, and why you would choose a particular protocol for several scenarios.
6. How does the performance of protocols like TCP change as the reliability of the connection varies? You could look at how the speed of getting data through changes if lots of packets need to be re-sent.

15.8. THE WHOLE STORY

For teachers

The OSI internet model is different from the TCP/IP model of the internet that Computer Scientists use to approach protocol design. OSI is considered and probably mentioned in the networking standards but the guide will use the computer science approach because it is simpler, however the main idea of layers of abstraction is more important to get across. You can read more about the differences here. (http://en.wikipedia.org/wiki/Internet_protocol_suite#Comparison_of_TCP.2FIP_and_OSI_layering)

Let's say I want to write an online music player. Okay, so I write the code for someone to press play on a website and the song plays. Do I now need to code up the protocol that streams the music? Fine, I write some UDP code. Now, do I need to go install the cables in your house? Sure, I jump in my van and spend a few weeks running cable to your house and make sure the packets can get over too.

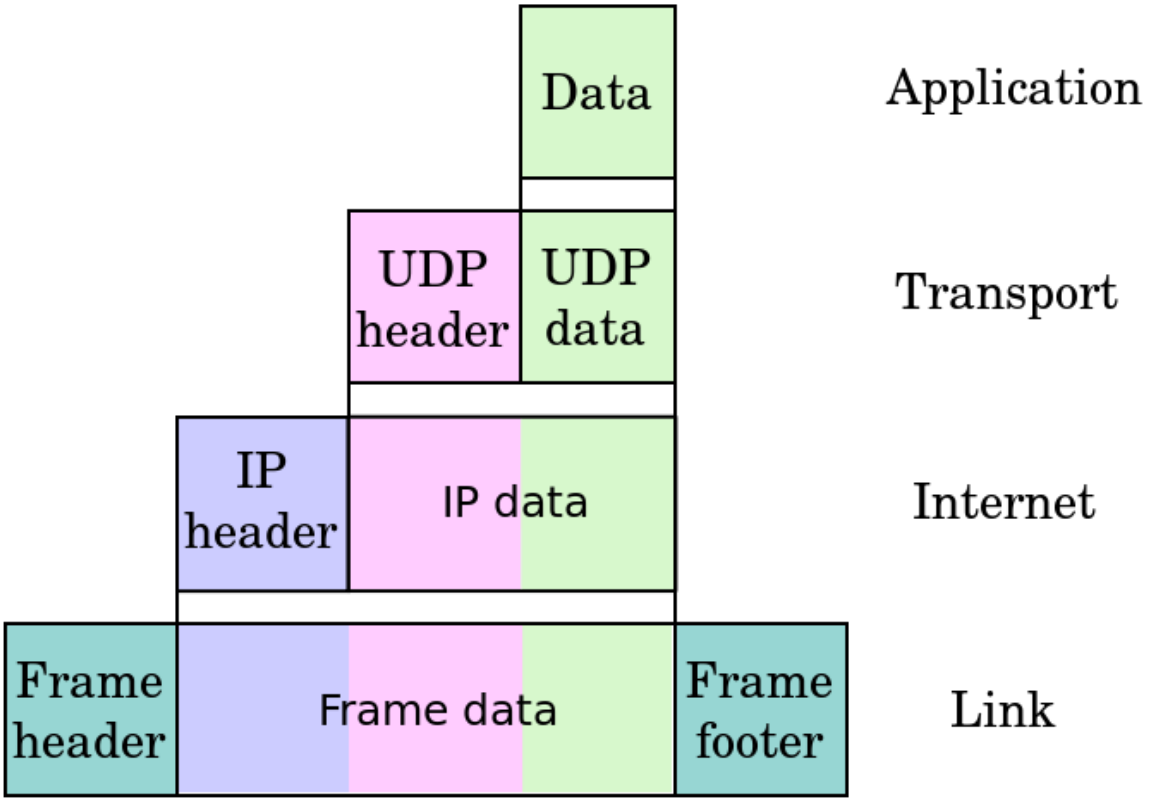
No. This sounds absurd. As a web developer, I don't want to worry about anything other than making my music player easy to use and fast. I *don't* want to worry about UDP and I *don't* want to worry about ethernet or cables. It's already done, I can assume it's take care of. And it is.

Internet protocols exist in layers. We have four such layers in the computer science internet model. The top two levels are discussed above in detail, the bottom two we won't focus on.The first layer is the Application Layer, followed by the Transport, Internet and Link layers.

At each layer, data is made up of the previous layers' whole unit of data, and then *headers* are added and passed down. At the bottom layer, the Link layer, a *footer* is added also. Below is an example of what a UDP packet looks like when it's packaged up for transport.

Jargon Buster

Footers and Headers are basically packet *meta-data*. Information about the information. Like a letterhead or a footnote, they're not part of the content, but they are on the page. Headers and Footers exist on packets to store data. Headers come before the data and footers afterwards.



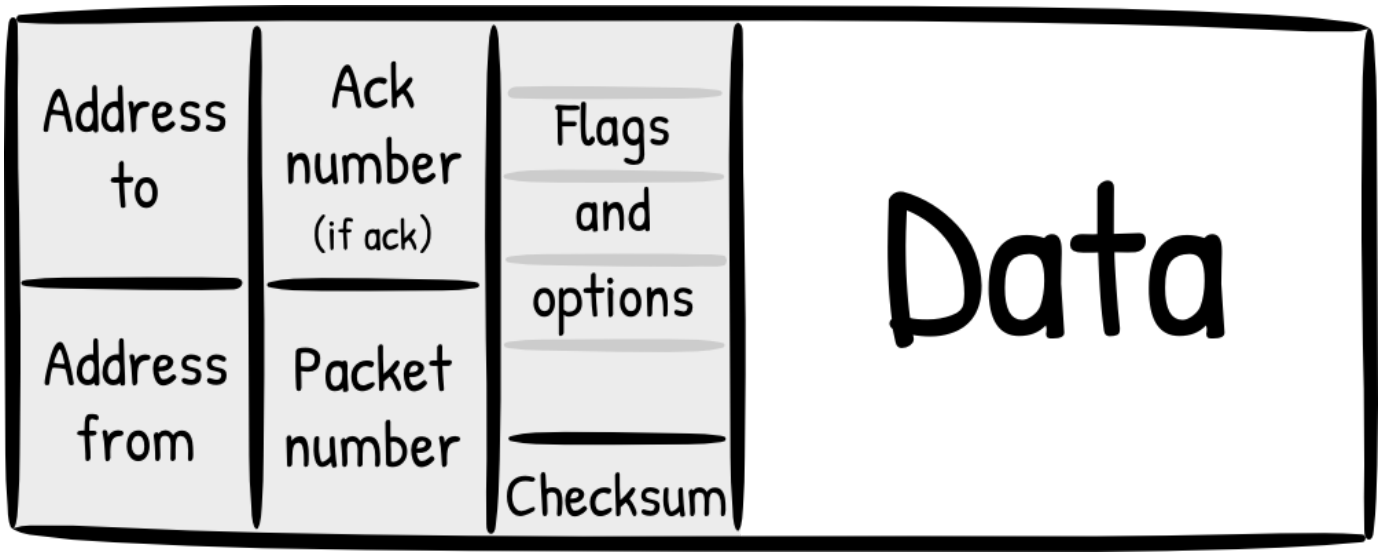
You can think of these protocols as a game of pass the parcel. When a message is sent in HTTP, it is wrapped in a TCP header, which is then wrapped in an IPv6 header, which is then wrapped in a Ethernet header and footer and sent over ethernet. At the other end, it's unwrapped again from an ethernet *frame*, back to a IP *packet*, a TCP *datagram*, to a HTTP *request*.

Jargon Buster

The name packet is a generic term for a unit of data. In the application layer units of data are called *data* or *requests*, in the transport layer, *datagram* or *segments*, in the Network/IP layer, *packet* and in the physical layer, a *frame*. Each level has its own name for a unit of data (segment, packet, frame, request etc), however the more generic "packet" is often used instead, regardless of layer.

This system is neat because each layer can assume that the layer above and below have guaranteed something about the information, and each layer (and protocol in use at that layer) has a stand-alone role. So if you're making a website you just have to program website code, and not worry about code to make the site work over wifi as well as ethernet. A similar system is in the postal system... You don't put the courier's truck number on the front of the envelope! That's take care of by the post company, which then uses a system to sort the mail and assign it to drivers, and then drivers to trucks, and then drivers to routes... none of which you need to worry about when you send or receive a letter or use a courier.

So what does a TCP segment look like?



As you can see, a packet is divided into four main parts, addresses (source, destination), numbers (sequence number, ANCK number if it's an acknowledgement), flags (urgent, checksum) in the header, then the actual data. At each level, a segment becomes the data for the next data unit, and that again gets its own header.

TCP and UDP packets have a number with how big they are. This number means that the packet can actually be as big as you like. Can you think of any advantages of having small packets? How about big ones? Think about the ratio of data to information (such as those in the header and footer).

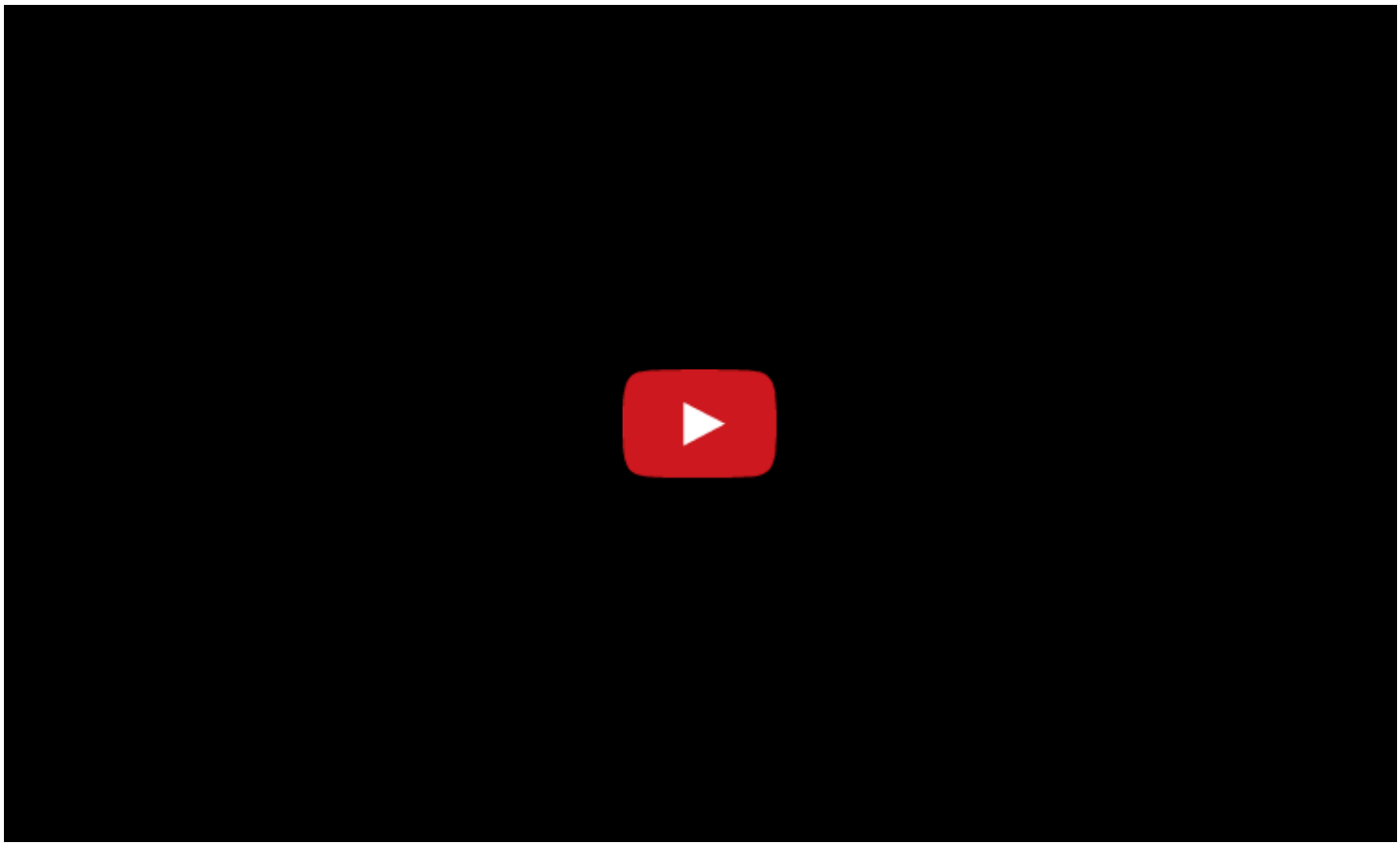
Curiosity

Here's an example of a packet trace on our network...(using tcpdump on the mac) (http://support.apple.com/kb/HT3994)

```
00:55:18.540237 b8:e8:56:02:f8:3e > c4:a8:1d:17:a0:d3, ethertype IPv4 (0x08
00), length 100: (tos 0x0, ttl 64, id 41564, offset 0, flags [none], proto
UDP (17), length 86)
  192.168.1.7.51413 > 37.48.71.67.63412: [udp sum ok] UDP, length 58
      0x0000:  4500 0056 a25c 0000 4011 aa18 c0a8 0107
      0x0010:  2530 4743 c8d5 f7b4 0042 1c72 6431 3a61
      0x0020:  6432 3a69 6432 303a b785 2dc9 2e78 e7fb
      0x0030:  68c3 81ab e28b fde3 cfef ae47 6531 3a71
      0x0040:  343a 7069 6e67 313a 7434 3a70 6e00 0031
      0x0050:  3a79 313a 7165
```

15.9. FURTHER READING

- The 2 generals problem is a famous problem in protocols to talk about what happens when you can't be sure about communication success. - http://en.wikipedia.org/wiki/Two_Generals%27_Problem (http://en.wikipedia.org/wiki/Two_Generals%27_Problem)
- What happens if you were to send packets tied to birds? IP over Avian Cariers - http://en.wikipedia.org/wiki/IP_over_Avian_Carriers (http://en.wikipedia.org/wiki/IP_over_Avian_Carriers)
- Protocols are found in the strangest of places.... Engine Order Telegraph - http://en.wikipedia.org/wiki/Engine_order_telegraph (http://en.wikipedia.org/wiki/Engine_order_telegraph)



15.9.1. EXTRA ACTIVITIES

- CS Unplugged Routing - Why do packets get delayed? http://csunplugged.org/routing-and-deadlock (http://csunplugged.org/routing-and-deadlock)
- Snail Mail - http://www.cs4fn.org/internet/realsnailmail.php (http://www.cs4fn.org/internet/realsnailmail.php)
- Code.org - The Internet https://learn.code.org/s/1/level/102 (https://learn.code.org/s/1/level/102)

15.9.2. USEFUL LINKS

- http://simple.wikipedia.org/wiki/TCP/IP (http://simple.wikipedia.org/wiki/TCP/IP)
- http://en.wikipedia.org/wiki/Internet_protocol_suite (http://en.wikipedia.org/wiki/Internet_protocol_suite)
- http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol (http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol)
- http://en.wikipedia.org/wiki/Internet_Relay_Chat (http://en.wikipedia.org/wiki/Internet_Relay_Chat)
- http://en.wikipedia.org/wiki/Transmission_Control_Protocol (http://en.wikipedia.org/wiki/Transmission_Control_Protocol)
- http://en.wikipedia.org/wiki/User_Datagram_Protocol (http://en.wikipedia.org/wiki/User_Datagram_Protocol)
- http://csunplugged.org/routing-and-deadlock (http://csunplugged.org/routing-and-deadlock)

Produced by Tim Bell, Jack Morgan, and many others based at the University of Canterbury (http://www.canterbury.ac.nz/), New Zealand.

Last updated on 16:01:27 New Zealand Daylight Time on Oct 16, 2014.Created using Sphinx (http://sphinx.pocoo.org/) 1.2b3.